eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

EUSKALCRAWLER III

Alumno: Casado Nieto, Xabier

Director: Pereira Varela, Juan Antonio

Curso: 2017-2018

Fecha: Bilbao, 17 de julio de 2018

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

Grado en Ingeniería Informática de Gestión y Sistemas de Información

Trabajo de Fin de Grado

EuskalCrawler III

Xabier Casado Nieto

dirigido por
Juan Antonio Pereira Varela

Bilbao, 17 de julio de 2018

Resumen

Castellano

Sistema online, desarrollado en Python y su framework web *Django*, capaz de recoger de forma automática información de páginas web y mediante el uso unificado de diversas APIs de eficacia comprobada, con el propósito principal de realizar un análisis de las tecnologías utilizadas en su creación, identificando en el proceso posibles vulnerabilidades.

Parte de una versión previa del proyecto, engloba todas las etapas del desarrollo de software e incluye un estudio de resultados en el que se analizan las principales tendencias en desarrollo de aplicaciones web a nivel local.

English

Online system, developed in Python and its web framework *Django*, capable of automatically collecting information from web pages and through the unified use of various APIs of proven effectiveness, with the main purpose of conducting an analysis of the technologies used in its creation, identifying potential vulnerabilities in the process.

It comes from a previous version of the project, encompasses all the stages of software development and includes a study of the results in which the main trends in web applications development are analyzed, at a local level.

Euskera

Python eta bere web framework *Django* erabiliz garatutako onlineko sistema, web orrialdeen informazioa automatikoki biltzeko gai eta egiaztatutako eraginkortasuna duten hainbat API batuz, bere sorreran erabilitako teknologien analisia egitea helburu nagusiarekin, prozesuan dauden ahultasun potentzialak identifikatuz.

Proiektuaren aurreko bertsio batetik dator, software garapenaren fase guztiak biltzen ditu eta emaitzak aztertzen ditu web aplikazioak garatzeko joera nagusiak aztertzen, tokiko mailan.

Prefacio

El proyecto *EuskalCrawler* surge inicialmente con el objetivo de automatizar la obtención de datos de empresas situadas en la CAV para posteriormente escanear sus páginas web, analizar cuáles están creadas partiendo de un sistema de gestión de contenidos (en inglés: *Content Management System*, más conocido por sus siglas CMS) y, en ese caso, detectar si son vulnerables.

Con el tiempo, el proyecto crece junto a la implementación de una interfaz de usuario web, que pone a disposición de las propias empresas una herramienta online para el escaneo de sus páginas webs — previa verificación de su autoría por parte de la empresa —, aportando por una parte información concreta de éstas y, al mismo tiempo, sirviéndose de esta colaboración por parte de las empresas para aumentar la base de datos global del proyecto.

El presente trabajo de fin de grado es la tercera fase del proyecto, desarrollado en *Django*, el más conocido framework web gratuito y open-source escrito en Python.

Al comienzo de la presente etapa, se partía de un sistema capaz de realizar el escáner mediante la combinación de los resultados obtenidos por las herramientas *Uniscan* y *Whatweb*. No obstante, el uso de un número tan reducido de escáneres, junto con la elección de éstos, conllevaba unos resultados finales insuficientes y poco robustos.

Es por ello que los dos hitos principales de esta actualización residen en **estandarizar/normalizar el formato de recogida de resultados**, por medio de un rediseño tanto del algoritmo como de la base de datos, para permitir aumentar el número de herramientas a utilizar en el escaneo y **analizar los resultados almacenados en la base de datos**, en busca de información estadística relevante del sector de desarrollo de aplicaciones web a nivel local, retomando así la idea original del proyecto.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

Es un hecho que avanzamos hacia una sociedad en la que las tecnologías facilitan día a día la creación, distribución y manipulación de la información, en ámbitos que abarcan desde lo social hasta lo económico. Cada vez tendemos a realizar más tareas cotidianas mediante Internet, bien sea por medio del ordenador, tablet o smartphone.

Entrando al ámbito local, cuatro de cada cinco personas son usuarias de Internet en Euskadi el presente año, según datos de *Eustat*[1] [Instituto Vasco de Estadística, 2018]. Este porcentaje abarca la totalidad de la franja de la población vasca comprendida entre los 15 y 54 años. Además, el 79,9 % de familias vascas dispone de conexión a Internet en sus viviendas, así como el 94,5 % dispone de teléfonos móviles.

Estos factores no son ajenos a las empresas, que tienen la necesidad de adaptarse a los tiempos que corren, lo cual buscan integrando las TIC en sus procesos y canales de comunicación: un estudio realizado el pasado año — nuevamente de la mano de *Eustat* —, reveló que el 51,4 % de empresas de Euskadi ya tienen presencia online [Instituto Vasco de Estadística, 2017]; dicho con otras palabras, más de la mitad del total de empresas vascas dispone de página web corporativa.

Queda claro que consumimos servicios web cada vez de una forma más habitual y masiva, así como la apuesta por parte de las empresas por las nuevas tecnologías, pero: ¿sabemos si lo realizamos de forma segura? ¿se encuentra a salvo la información que manejan los negocios por medio de estos nuevos canales? Es aquí donde entra el concepto de la seguridad informática.

Desde el punto de vista de la empresa, la seguridad informática la conforman todas las medidas y mecanismos encargados de que los recursos de su sistema se utilicen de la manera en que se decidió y que el acceso a la información allí contenida, así como su modificación, solamente sea posible

[1] Página oficial del Instituto Vasco de Estadística (*Eustat*): `http://www.eustat.eus/`

a las personas que se encuentren acreditadas y dentro de los límites de la autorización [Costas, 2011]. Por ende, se considera un sistema seguro aquel que cumple con los prinicipios de Confidencialidad, Integridad, Disponibilidad, Autenticación y No-repudio (CIDAN).

Si bien las empresas son conscientes de la importancia de ser visibles en Internet y han dado pasos en esta dirección, la seguridad informática no es tan tenida en cuenta en esta ecuación; muchas veces, estas medidas suelen llegar demasiado tarde. Sin ir más lejos, el segundo cuatrimestre del pasado año abría con el titular: *"El Gobierno confirma un ciberataque masivo a empresas españolas"* [Muñoz, 2017] ¿la causa? equipos informáticos desactualizados, provocando una vulnerabilidad en el sistema operativo de Microsoft de la que el malware se valía; falta de conocimiento o de inversión de capital.

Resulta de vital importancia conocer el software que se está utilizando, para analizar sus agujeros de seguridad, así como mantenerlo actualizado a fin de prevenirlos. Existen mecanismos que pueden ser de gran ayuda en esta tarea: la lista CVE (del inglés: *Common Vulnerabilities and Exposures*) registra y proveé en su página web[2] información actualizada sobre vulnerabilidades de seguridad conocidas, en la que cada referencia tiene un número de identificación único, ofreciendo así una nomenclatura común y pública para el problema que nos concierne y en el que se basa el presente trabajo de fin de grado, en adelante TFG.

1.1. Origen y descripción del proyecto

Teniendo en cuenta todo lo comentado previamente surge *EuskalCrawler*, cuya idea original data de 2015, año en el cual se comenzó a desarrollar la que fue la primera versión del proyecto, en base a un concepto ideado por el docente Juan Antonio Pereira Varela, miembro del departamento de Lenguajes y Sistemas Informáticos (LSI), y cuyo desarrollo corrió a cargo del alumno David López Angulo [López, 2016]. La idea consistía en un sistema capaz de recopilar de forma automática información de empresas vascas, con el propósito principal de obtener sus páginas web y realizar un análisis de las tecnologías utilizadas en su creación, principalmente si la web estaba realizada a partir de un sistema de gestión de contenidos — en inglés: *Content Management System (CMS)* —, identificando en el proceso posibles vulnerabilidades.

[2]Página oficial de la *MITRE CVE List*: `http://cve.mitre.org/`

Euskal Crawler

En esta primera versión, y tras desestimarse la idea de implementar un sistema de *crawling* (si bien su nombre bautizó al proyecto) en pro del uso de la técnica *scraping*, se partió de un listado público de empresas de la CAV para obtener una lista de sus respectivas páginas web mediante la utilidad online *import.io*[3], con las que formar una primera base de datos. Una vez finalizado el proceso por el cual se obtenían y almacenaban las páginas web, un script se encargaba de analizarlas de forma individual mediante la herramienta *wig*[4], obteniendo información técnica para cada una de ellas: tecnologías utilizadas, versión de éstas y posibles vulnerabilidades asociadas a esta tupla. Toda la programación se realizó en Python 3.

Adicionalmente, se implementó una interfaz de usuario web, en la cual se podía introducir una URL en particular para realizar el escaneo de ésta, previo registro. El sitio web mostraba además gráficas, en base a los resultados del análisis del listado de webs almacenadas en la base de datos. Esta parte se implementó utilizando PHP como lenguaje backend.

El concepto había quedado plasmado, pero con una puesta en marcha algo pobre: el proyecto consistía en una interfaz gráfica simple mediante la cual presentar al usuario los resultados obtenidos con una única herramienta — asumiendo que *wig* bastaba para este cometido — y almacenados en una base de datos que, si bien recopilaba información relevante de un elevado número de empresas, no mostraba al usuario más que una pequeña parte del conjunto total.

Luego de la defensa de esta versión de la aplicación, entró en juego una primera actualización, de la mano del alumno Iñigo Gallego González. Éste exponía en su memoria [Gallego, 2018] cómo el resultado inicial había sido desechado y la necesidad de un rediseño, mejorando el concepto y tomando como punto de partida el código ya disponible.

Un detector de vulnerabilidades de sitios web

El alumno introdujo en esta etapa una serie de mejoras en la interfaz de usuario online, que partieron de un rediseño mediante *Django* [5], el conocido framework web de Python. Tras quedar definido el sector de usuarios

[3] *import.io*: `https://www.import.io/`
[4] *wig*: `https://github.com/jekyc/wig`
[5] Página oficial de *Django*: `https://www.djangoproject.com/`

al que ésta iba destinada — empresarios que buscaban saber si sus páginas web presentabas problemas de seguridad —, se aplicaron las medidas de seguridad pertinentes de las que carecía la versión inicial, que pasaban desde la confirmación de registro de los usuarios por correo electrónico a la verificación de la autoría de las webs a escanear por parte de estos. Este fue el punto fuerte y eje sobre el que radicó la actualización de Iñigo, acorde a la cual renombró el proyecto a: *EuskalCrawler, un detector de vulnerabilidades de sitios web.*

En lo que respecta al algoritmo y tras un trabajo de investigación, se eliminó la herramienta utilizada en la primera versión por problemas de compatibilidad, sustituyéndola por dos nuevas, con cometido similar: *Uniscan*[6] y *Whatweb*[7]. Cabe mencionar también que en esta etapa no se utilizó una nueva base de datos de empresas, sino que se realizó un nuevo escaneo completo de las webs ya registradas, almacenando los resultados obtenidos por las dos nuevas herramientas.

Es precisamente aquí cuando queda latente el principal error de esta versión: su base de datos. En lugar de la implementación de un nuevo modelo relacional, se añadieron a las ya existente las tablas necesarias para cada una de las herramientas añadidas, sin comprobar si ya existía una entidad para satisfacer cada problema, conllevando a redundancia de información y mal uso de la ya almacenada (por ejemplo, esta actualización eliminó la muestra de resultados finales en gráficas). Nos encontramos con, en el mejor de los casos, el triple de información recogida en comparación al proyecto inicial, en lo que resulta una base de datos no normalizada y que únicamente muestra una pequeña parte de la información almacenada al usuario solicitante del escaneo, suponiendo que la web proceda de un escaneo particular y no a la lista de páginas escaneadas de forma automática. A esto hay que añadir también la falta de un histórico de resultados.

A su finalización, ya en la segunda etapa, no había logrado desarrollarse el proyecto en su totalidad; si bien se había avanzado de forma notable en la interfaz gráfica, haciendo especial hincapié el ámbito de la seguridad, únicamente se había pasado de utilizar una herramienta en el escaneo de los sitios web a dos y uno de los puntos de los que se presumía en la memoria final era precisamente una de sus mayores carencias: no resultaba una tarea fácil la inclusión de nuevas herramientas con las que complementarlo.

[6] *Uniscan*: `https://github.com/poerschke/Uniscan`
[7] *Whatweb*: `https://github.com/urbanadventurer/WhatWeb`

1.2. Situación del proyecto

Expuesto lo anterior, Juanan — que había ejercido de director del proyecto en sus dos fases — veía necesaria una vez más la continuación del proyecto, a fin de mejorarlo para ajustarlo a la que era su idea inicial; la elección de las herramientas utilizadas para el escaneo de los sitios web y el número tan reducido de éstas concluyeron en unos resultados finales poco robustos e insuficientes.

EuskalCrawler III

Bajo esta premisa, el presente TFG, que es a su vez la tercera fase de *EuskalCrawler*, busca principalmente la consecución de dos logros.

Por un lado, ampliar el número de herramientas utilizadas, estandarizando tanto su uso por los componentes de la aplicación como el formato de salida de la información devuelta por ellas. Esto incluye, por ende, un rediseño del modelo relacional actual, eliminando toda redundancia, relación espuria o ausencia de ésta y tablas en desuso en el proceso, previa definición de las entidades imprescindibles. Huelga decir que, tras este primer punto, habrá que adaptar el algoritmo encargado de la tarea principal del proyecto, manteniendo la sincronía con los cambios en el modelo.

La segunda meta a alcanzar consiste en retomar el análisis y muestra, de forma simple y clara, de los resultado obtenidos en base al escaneo masivo de webs de empresas en territorio vasco, a fin de poner en conocimiento de los usuarios las conclusiones fruto del presente trabajo.

Al tomar como punto de partida la versión previa, se utilizará nuevamente el framework web *Django*.

1.3. Comparativa entre versiones

A continuación, expuestas las diferentes fases del proyecto, se va a proceder a la realización de una comparativa entre la versión actual y sus dos versiones precedentes, puesto que se considera innecesario un análisis de antecedentes, luego de haberse realizado dicho estudio previamente y en dos ocasiones, encontrándose éste disponible en las memorias elaboradas por los alumnos encargados de las fases primera y segunda del proyecto [López, 2016, Gallego, 2018].

En la Tabla 1.1 se resumen, de forma superficial, las principales diferencias entre las distintas etapas y las carencias que la presente versión del proyecto pretende cubrir; sirva ésta como precedente para los consecutivos capítulos, en los que se profundizará debidamente en estos aspectos, bien a la hora de establecer los objetivos, así como en la posterior Especificación de Requisitos de Software.

Primera versión	***Segunda versión***	***EuskalCrawler III***
Registro y autenticación de usuarios	Registro con confirmación y autenticación de usuarios	Registro con confirmación y autenticación de usuarios por nombre o email
Un único escaneador	Dos escaneadores sin formato estandarizado	Aumento sustancial del número de escaneadores con formato de salida de resultados y almacenamiento estandarizados
Base de datos para el almacenamiento de resultados ceñido al único escaneador	Sin definición de un nuevo modelo de base de datos, carencia de relaciones y estructura	Rediseño del modelo de base de datos
Implementación de la técnica de scraping para el almacenamiento de información de empresas en base a listado online	Implementación de un script para el escaneo inicial de las empresas almacenadas	Implementación de una tarea automatizada para el escaneo periódico de empresas almacenadas
Escáner de sitios web sin verificación de autoría	Verificación de autoría implementada	Verificación de autoría implementada y automatizada
Muestra de resultados en un único gráfico	Sin gráficos de resultados	Muestra de resultados en diversos gráficos actualizados regularmente
Interfaz de usuario web insuficiente	Interfaz de usuario web no validada para diferentes plataformas	Interfaz de usuario web multiplataforma y diseño responsivo

Tabla 1.1: Comparativa entre versiones

1.4. Motivaciones para la elección del proyecto

Hablando en esta sección desde un punto de vista personal, subjetivo y, por ende, en primera persona, varias son las razones por las que acepté llevar a cabo el rediseño del proyecto por encima de otras propuestas de TFG, las cuales voy a tratar de exponer a continuación.

En primer lugar, la elección del proyecto estaba condicionada a la rama de la informática hacia la que siento más cercanía y sobre la cual quería sustentar su desarrollo; previo a mi incursión en el Grado en Ingeniería Informática de Gestión y Sistemas de Información, había cursado el CFGS[8] de Desarrollo de Aplicaciones Web, así como adquirido experiencia en el ámbito laboral (y realizado pequeños proyectos) en este campo. Desde un principio, tenía claro que el desarrollo web era la dirección hacia la cual quería enfocar mi TFG.

De acuerdo a lo anterior y habiendo cursado la asignatura optativa de Desarrollo de Aplicaciones Web Enriquecidas, impartida por Juanan, encontré en él a la persona idónea para realizar las funciones de dirección. Más aún tras darme a conocer el proyecto del que trata el presente documento, del cual me indicó que ya se encontraba comenzado y estaba desarrollado en una tecnología y lenguaje muy demandados por las empresas en actualidad como lo son Django y Python, lo cual me animó a aceptarlo de forma casi inmediata.

Personalmente, considero que trabajar en un proyecto que ya ha sido comenzado por terceras personas supone un reto que se ajusta bastante a situaciones reales en un entorno laboral de este sector: labores como la interpretación y comprensión de código de otros desarrolladores, análisis de puntos fuertes y carencias, son problemas que surgen en el día a día y a los que tenemos que habituarnos, a fin de solventarlos en el menor tiempo posible. Es por esto, que otro de los puntos que valoré positivamente fue el hecho de tener que realizar el rediseño y ampliación de contenidos de la presente aplicación.

La integración en el proyecto con no sólo una interfaz de programación de aplicaciones (del inglés, *Application Programming Interface (API)*), sino varias, pone de manifiesto y concuerda con mi opinión de rechazo a la expresión *"reinventar la rueda"* y es que, si el problema ya está solucionado,

[8] Ciclo Formativo de Grado Superior

resulta más conveniente y rápido integrar la solución para la que ya se ha demostrado validez que crear una partiendo de cero; más si, como en este caso, las soluciones disponibles son además varias y nos permiten la libertad de escoger las que más se adecúen a nuestras necesidades. Otro tanto para la propuesta, por el hecho de enlazar a herramientas de escaneo de terceros, además de estar en sintonía con lo comentado en el párrafo anterior.

Por último, uno de los motivos de mayor peso: de acuerdo a lo expuesto, el proyecto realmente tiene utilidad para otras personas y sus resultados son visibles por medio de datos estadísticos. Este trabajo promete dar sus frutos, ya sea beneficiando a las empresas que utilicen la herramienta a fin de conocer si su sitio web es vulnerable, como también en forma de gráficos que nos acerquen al estudio de las tendencias en desarrollo web a nivel local.

2. Planteamiento inicial

Esta sección forma, junto con la anterior introducción, el Documento de Objetivos de Proyecto (DOP) y tiene como fin exponer los objetivos y alcance, así como un análisis con el método de trabajo, la planificación temporal y estimación económica del proyecto, ilustrando su arquitectura y describiendo las herramientas que se van a utilizar en su elaboración. Se reparará también en los posibles riesgos para una gestión eficiente de los mismos.

2.1. Objetivos

Como bien se ha comentado previamente, esta tercera fase del proyecto busca la consecución principal de dos logros, los cuales se pueden definir como:

- **estandarizar/normalizar el formato de recogida de resultados**, por medio de un rediseño tanto del algoritmo como de la base de datos, para permitir aumentar el número de herramientas a utilizar en el escaneo.
- **analizar los resultados almacenados en la base de datos**, en busca de información estadística relevante del sector de desarrollo de aplicaciones web a nivel local, retomando así la idea original del proyecto.

Un tercer objetivo, en este caso a tener en cuenta a la hora de realizar la planificación temporal, será el **establecimiento de un *deadline* de seis meses**, a fin de agilizar el proyecto, así como no alargar en exceso la finalización del presente grado universitario. La fecha de finalización será la que establezca la entrega del TFG más cercana a dicho plazo (*TERCERA CONVOCATORIA: Solicitud de defensa a través de GAUR y carga del TFG en la aplicación ADDI del estudiante: del 28 al 29 de junio de 2018*)[1].

[1] *Trabajo de Fin de Grado - curso 2017/18*: `https://www.ehu.eus/es/web/ingeniaritza-bilbo/graduen-araudiak`

Si bien tenemos definidos dos hitos principales, el proyecto también se ramifica en dos partes, diferenciables entre sí y en base a las que se va a dividir esta sección. A continuación, se expondrá cada una de ellas, junto con los objetivos definidos y que han de cumplimentarse en cada caso, así como objetivos ya satisfechos en versiones previas del proyecto y que han de mantenerse en la presente.

2.1.1. Captura, almacenamiento y análisis de resultados

Proceso automático por el cual, y periodicamente, se escanean las páginas web almacenadas en la base de datos. Se utilizarán los resultados obtenidos para realizar una fotografía — con datos estadísticos y utilizando como tamaño de la muestra el total de webs escaneadas —, de las tendencias en desarrollo web a nivel local.

- **Rediseño de la base de datos**, a fin de almacenar los resultados de forma estandarizada.
- **Rediseño del algoritmo de escaneo**, en busca de adaptar éste a la base de datos final.
- **Realización de un escaneo inicial** y sin errores de todas las páginas almacenadas hasta la fecha, así como **realización de un nuevo escaneo** de las 50 páginas más consultadas **con una periodicidad mensual**.
- **Integración dinámica de cualquier nueva API**, así como elección dinámica y sin necesidad de reescritura de código de las herramientas a utilizar en el escaneo.
- **Creación de estadísticas**, en base a todos los sitios web analizados, en forma de gráficos. Se actualizarán con una periodicidad semanal, reflejando fielmente el estado de la base de datos.

2.1.2. Interfaz de usuario web

Sitio web online y accesible desde el navegador, que implementa una interfaz gráfica amigable con el usuario, desde la cual un empresario puede registrarse a fin de almacenar su página web — o una lista de ellas — y realizar o repetir un escaneo, previa verificación de su autoría.

- Modificación del diseño actual y **conversión a un diseño responsivo**.

- Muestra al empresario de la información referente a resultados de un escaneo de forma **limpia y ordenada por secciones**.
- Sistema de **confirmación de correo eléctronico**, necesario para la activación de la cuenta registrada y la autenticación en la aplicación.
- **Verificación de la autoría** de cada página web almacenada en la cuenta del empresario, mediante un sistema que comprueba que dicho usuario puede añadir los contenidos solicitados por la aplicación a la página en cuestión y sin la cual no se permite el escaneo del sitio.
- **Presentación visual** al público en general — aún sin necesidad de autenticación previa — en forma **de gráficos estadísticos**, en base a todos los sitios web analizados.

2.2. Método de trabajo

El proyecto va a constar de una única entrega, en la cual se presentará la aplicación final. No obstante, se acordarán reuniones puntuales con el director del proyecto, para comentar los avances así como los posibles problemas que puedan surgir y tratar de solventarlos, además de decidir los casos de uso que serán necesarios incluir en la aplicación y los que no.

De cara al tiempo incurrido en la elaboración del proyecto, se establece en base a disponibilidad y bajo criterio personal, una jornada de trabajo del orden de 4 horas diarias para los días laborales y 8 horas de trabajo diarias para los fines de semana; el cómputo total asciende a 36 horas semanales.

2.3. Herramientas

En esta sección se explicarán de forma muy breve las herramientas a utilizar a lo largo de las diferentes etapas de desarrollo del proyecto. A modo de aclaración previa, esta sección no tiene como fin detallar las tecnologías utilizadas para la implementación, que se tratarán posteriormente a lo largo del presente documento y una vez finalizada la planificación inicial.

- **Bitbucket**: servicio de alojamiento basado en web, escrito en Python, para proyectos que utilizan el sistema de control de versiones Mercurial y Git. Permite un número limitado de repositorios privados.

- **Cacoo**: herramienta online para hacer diagramas de diversos tipos, desde el esquema de una oficina hasta diagramas UML, pasando por el prototipado de pantallas.
- **Dia**: programa para la elaboración de diagramas estructurados; está concebido de forma modular, con múltiples paquetes, cada cual con una colección de formas, para abarcar diferentes necesidades.
- **Git**: sistema de gestión de versiones de código abierto, diseñado por Linus Torvalds. Está integrado en sistemas como GitHub, Bitbucket o GitLab y se puede usar en casi todo tipo de plataformas.
- **MySQL Workbench**: herramienta que proporciona una interfaz gráfica de usuario para el diseño de bases de datos, integrando desarrollo de software, administración de bases de datos, diseño de bases de datos, gestion y mantenimiento, para el sistema de base de datos MySQL.
- **Overleaf**: herramienta de escritura y publicación colaborativa en línea de LaTeXy Rich Text. Con ella, se puede ver en tiempo real el documento de texto enriquecido resultante del código que se escribe.
- **plot.ly**: se trata de una API para Python que facilita la creación de gráficos, generándolos en base a los datos definidos por el usuario y mostrándolos mediante una interfaz gráfica interactiva accesible desde un navegador.
- **PyCharm**: es un entorno de desarrollo integrado que permite escribir código en Python, HTML y CSS y además incluye herramientas para la compilación y comprobación del funcionamiento del código.
- **Visual Paradigm**: herramienta que permite realizar y exportar a imagen diagramas UML.

2.4. Arquitectura

Para definir el alcance del proyecto, conviene contemplar previamente cuáles van a ser los equipos físicos con los que vamos a contar, así cómo la división a nivel de subsistemas lógicos que se va a realizar.

En el caso que aplica a este proyecto, estas especificaciones ya han sido establecidas en la versión previa a la presente; no obstante, se considera conveniente completar la documentación de esta sección.

2.4.1. Arquitectura física

El proyecto sigue la clásica arquitectura web de cliente/servidor, específicamente con cliente ligero, recayendo el grueso de la lógica de negocio de la aplicación en el servidor. En lo que a este respecta, una misma máquina albergará el servidor web y así como el de base de datos, la cual utilizará un sistema operativo de la familia *Linux*.

2.4.1.1. Cliente

Será el encargado de mostrar la información de una forma estructurada y fácilmente entendible para el usuario final, por medio de un navegador. Por tanto, se entiende por cliente cualquier dispositivo capaz de establecer una conexión a Internet para la realización de dicho fin, bien sea un teléfono móvil, tablet o equipo portátil o de sobremesa.

2.4.1.2. Servidor

Como bien se ha comentado previamente, el servidor será la máquina a la cual se realizarán las peticiones de los recursos a mostrar en el navegador del equipo cliente. Albergará todos los ficheros que implementan la lógica funcional de la aplicación (gestionando la interactividad del usuario con ésta y realizando el acceso a datos), así como la información almacenada por la propia aplicación y contenida en una base de datos.

2.4.2. Arquitectura lógica

Al desarrollarse la aplicación utilizando el framework web Django, éste es quien impone la arquitectura lógica: concretamente de tres capas, mediante el uso de su patrón de diseño *Model-Template-View (MTV)*, el cual no es más que una interpretación libre, de la mano del equipo de desarrolles de este framework, del conocido patrón de diseño Modelo-Vista-Controlador

(MVC) [Holovaty y Kaplan-Moss, 2009].

Este patrón se divide, al igual que su referente, en tres capas:

- La *M* proviene del **modelo**/s (*models*), la capa de persistencia de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a los mismos, cómo validarlos, cuál es el comportamiento o fin y las relaciones entre los datos.
- La *T* hace referencia a las **plantillas** (*templates*), la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: cómo se muestra la información sobre una página web.
- La *V* la conforman las **vistas** (*views*), la capa de la lógica de negocios, conocida en otras ocasiones como controlador. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada, sirviendo de nexo entre ambos.

Adicionalmente, la resolución de **solicitudes** (*request*) de recursos por parte del usuario se gestiona mediante el *urlresolver* de Django, tomando como referencia una serie de patrones — definidos en la implementación de la propia aplicación — y llamando a la función del fichero *views* definida para el patrón que coincide con la petición del usuario. Es dicha función la que se encargará de **responder** (*response*) al usuario con la *template* pertinente, mostrando los datos requeridos obtenidos del *model*.

Un resumen de la arquitectura, así como del procesamiento de una solicitud, se puede observar en la Figura 2.1.

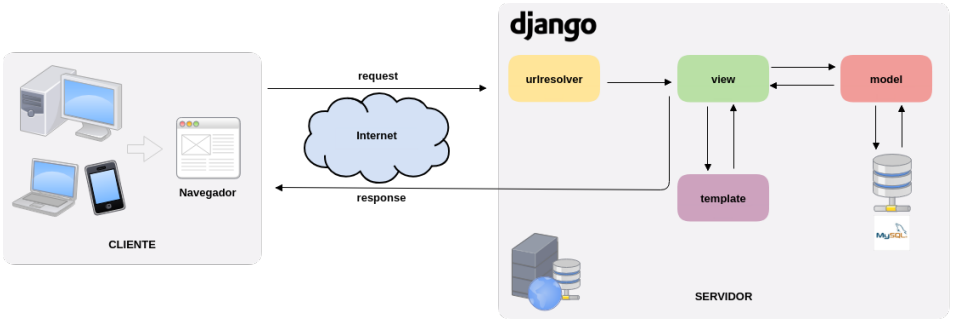


Figura 2.1: Arquitectura y procesamiento de solicitudes

2.5. Alcance

Para lograr la consecución de los objetivos establecidos, resulta de vital importancia el realizar una correcta división de tareas a completar. En este caso, la forma de definir el alcance se va a reflejar mediante una Estructura de Descomposición del Trabajo (EDT).

La distribución se va a realizar en forma de una serie de módulos cuyo orden de realización es, en su mayoría, intercambiable (prevaleciendo en todo caso las precedencias y siempre que éstas no establezcan lo contrario) y que se corresponden a los mostrados en la Figura 2.2.

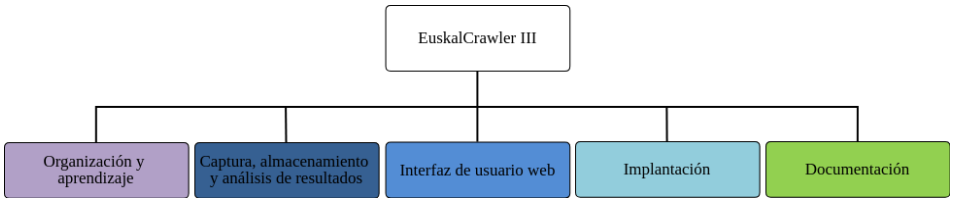


Figura 2.2: Tareas principales de la EDT

Esta decisión se ha tomado teniendo en cuenta que, a pesar de la interdependencia de los diferentes módulos, éstos pueden desarrollarse de forma aislada: por un lado debido a la naturaleza de la arquitectura elegida y, por otra parte, que podemos realizar los diferentes módulos tanto de forma pareja como alterando el orden, sin que esto suponga diferencia alguna en el resultado final. De hecho, en un proyecto de mayor escala, estos módulos podrían llevarse a cabo por distintos equipos de trabajo.

Adicionalmente, este tipo de distribución de tareas induce a la realización de forma individual de las distintas fases correspondiente al ciclo de desarrollo del proyecto tales como la captura de requisitos, análisis y diseño de cada módulo, frente a un único diseño integrado. Por ende, los diagramas resultantes de esta definición del alcance serán más sencillos de realizar e interpretar, facilitando del mismo modo el tedioso trabajo que conlleva su realización.

La realización del proyecto seguirá una estrategia *bottom-up*, o de abajo hacia arriba: conociendo el conjunto de módulos individuales a realizar, estos se van juntando en grupos, hasta conformar la suma el proyecto en

su totalidad.

El siguiente es un breve detalle de los módulos, de forma introductoria al posterior desglose de éstos:

- **Organización y aprendizaje**: comprende todas las tareas relacionadas con la organización del proyecto, así como el diseño del DOP y estudio de las diferentes herramientas y lenguajes que se van a utilizar.

- **Captura, almacenamiento y análisis de resultados**: engloba la captura de requisitos, análisis, diseño, implementación y desarrollo de las pruebas del backend o entorno servidor de la aplicación.

- **Interfaz de usuario web**: en sintonía con el anterior módulo, se corresponde a todas las fases del desarrollo del frontend o entorno cliente de la aplicación.

- **Implantación**: hace referencia al despliegue en servidor y configuración del acceso a la aplicación.

- **Documentación**: redacción de todo tipo de material necesario para explicar el completo desarrollo de la aplicación, así como el conjunto de manuales para la correcta instalación y utilización de la misma.

A continuación se presentará el desglose de tareas para cada uno de los módulos principales, junto con la explicación de cada una de ellas. Se ha optado por asignar una duración estimada pesimista, de cara a cubrir posibles contratiempos en la elaboración de las tareas para su posterior planificación temporal.

2.5.1. Organización y aprendizaje

Conjunto inicial de acciones a llevar a cabo cuyo propósito es la definición de las tareas necesarias para la elaboración del proyecto, así como la formación en las tecnologías que lo requieran. Alberga a su vez un submódulo, encargado de definir las tareas asociadas a la elaboración del DOP.

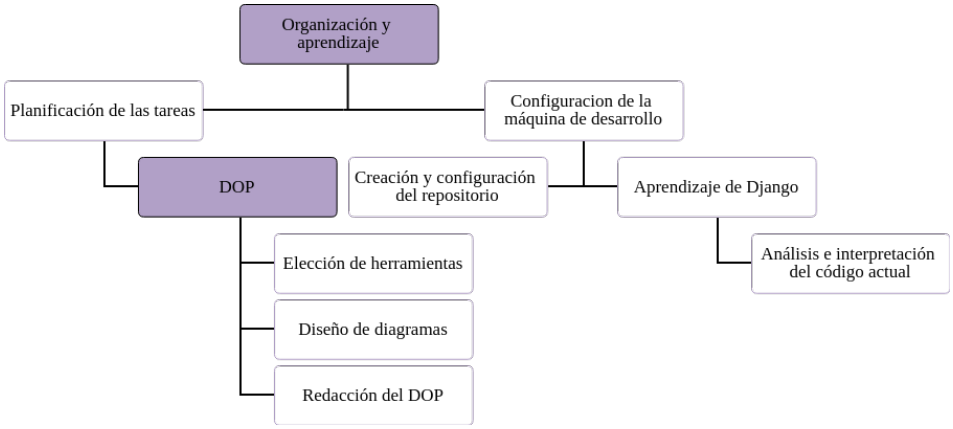


Figura 2.3: EDT correspondiente al módulo de Organización y aprendizaje

Planificación de las tareas
Paquete de trabajo: Organización y aprendizaje. **Duración estimada**: 8 horas.
Descripción: Identificar, definir y estimar la duración de las tareas que se tienen que realizar para la elaboración del TFG. **Salidas/Entregables**: EDT. **Recursos necesarios**: Software para diseño de planificación de proyectos.

Tabla 2.1: Planificación de las tareas

Configuración de la máquina de desarrollo
Paquete de trabajo: Organización y aprendizaje. **Duración estimada**: 2 horas.
Descripción: Configuración del entorno de desarrollo mediante la instalación de un entorno de desarrollo integrado y dependencias necesarias. **Salidas/Entregables**: Entorno de desarrollo preparado. **Recursos necesarios**: Dependencias y herramientas requeridas o heredadas de versiones previas de la aplicación.

Tabla 2.2: Configuración de la máquina de desarrollo

Creación y configuración del repositorio
Paquete de trabajo: Organización y aprendizaje. **Duración estimada**: 2 horas.
Descripción: Creación de un repositorio en Bitbucket, subida del código de la versión previa de la aplicación y vinculación de éste con el entorno de desarrollo integrado (IDE) de la máquina de desarrollo. **Entradas**: Entorno de desarrollo preparado. **Recursos necesarios**: Entorno de desarrollo integrado, sistema de control de versiones, código fuente de la versión previa y navegador.

Tabla 2.3: Creación y configuración del repositorio

Aprendizaje de Django
Paquete de trabajo: Organización y aprendizaje. **Duración estimada**: 72 horas (2 semanas).
Descripción: Realización de un curso online para el aprendizaje desde cero del framework web Django y, al mismo tiempo, de Python. **Entradas**: Entorno de desarrollo preparado. **Recursos necesarios**: Entorno de desarrollo integrado, curso online *Tango with Django* [Azzopardi y Maxwell, 2016], documentación oficial de Django[2], tutorial de aprendizaje de Python en 10 minutos[3], entorno de desarrollo integrado

Tabla 2.4: Aprendizaje de Django

[3] *Documentación oficial de Django*: `https://www.djangoproject.com/`

[3] *Tutorial - Learn Python in 10 minutes*: `https://www.stavros.io/tutorials/python/`

Análisis e interpretación del código actual
Paquete de trabajo: Organización y aprendizaje. **Duración estimada**: 20 horas.
Descripción: Analizar el código actual actual de la aplicación, junto con su documentación, a fin de entender qué y cómo se está haciendo y decidir lo que ha de añadirse y eliminarse. **Precedencias**: Aprendizaje de Django. **Entradas**: Entorno de desarrollo preparado. **Recursos necesarios**: Entorno de desarrollo integrado, código y documentación de la versión previa.

Tabla 2.5: Análisis e interpretación del código actual

2.5.1.1. Documento de Objetivos de Proyecto

El Documento de Objetivos de Proyecto (DOP) comprende la introducción y el planteamiento inicial de la memoria del TFG; su totalidad se extiende a las secciones descripción, objetivos, herramientas, arquitectura, método de trabajo, alcance, gestión de riesgos, planificación temporal y evaluación económica, suponiendo el punto de partida del proyecto. Se ha optado por la siguiente división de tareas para su realización.

Elección de herramientas
Paquete de trabajo: Documento de Objetivos de Proyecto. **Duración estimada**: 2 horas.
Descripción: Recopilar información sobre las herramientas necesarias para la realización de la totalidad de tareas del proyecto. Se parte con ventaja, ya que se dispone de la documentación de las versiones previas, donde consultar por qué herramientas se optó en su día para la elaboración de las diferentes tareas. **Entradas**: EDT. **Salidas/Entregables**: Lista de herramientas a utilizar. **Recursos necesarios**: Internet y documentación de las versiones previas como referencia, experiencia previa.

Tabla 2.6: Elección de herramientas

Diseño de diagramas
Paquete de trabajo: Documento de Objetivos de Proyecto. **Duración estimada**: 4 horas.
Descripción: Creación de los diagramas necesarios para incluir en el DOP, no contemplando aquellos para realizar tareas posteriores. **Entradas**: EDT y lista de herramientas a utilizar. **Salidas/Entregables**: Diagramas para el DOP **Recursos necesarios**: Software para diseño de planificación de proyectos.

Tabla 2.7: Diseño de diagramas

Redacción del DOP
Paquete de trabajo: Documento de Objetivos de Proyecto. **Duración estimada**: 28 horas.
Descripción: Por simplificar el desglose del DOP, esta única tarea incluye la redacción de las secciones descripción, objetivos, herramientas, arquitectura, método de trabajo, alcance, gestión de riesgos, planificación temporal y evaluación económica. **Entradas**: EDT, lista de herramientas a utilizar y diagramas para el DOP. **Salidas/Entregables**: Documento de Objetivos de Proyecto (DOP). **Recursos necesarios**: Editor de textos y software para diseño de planificación de proyectos.

Tabla 2.8: Redacción de DOP

2.5.2. Captura, almacenamiento y análisis de resultados

Engloba todas las tareas necesarias para realizar las diferentes fases del desarrollo del backend o entorno servidor de la aplicación, como lo son la captura de requisitos, análisis y diseño, implementación y desarrollo de las pruebas. Con respecto a la parte de la implementación, se ha divido a su vez en diferentes tareas, que pueden realizarse en cualquier orden.

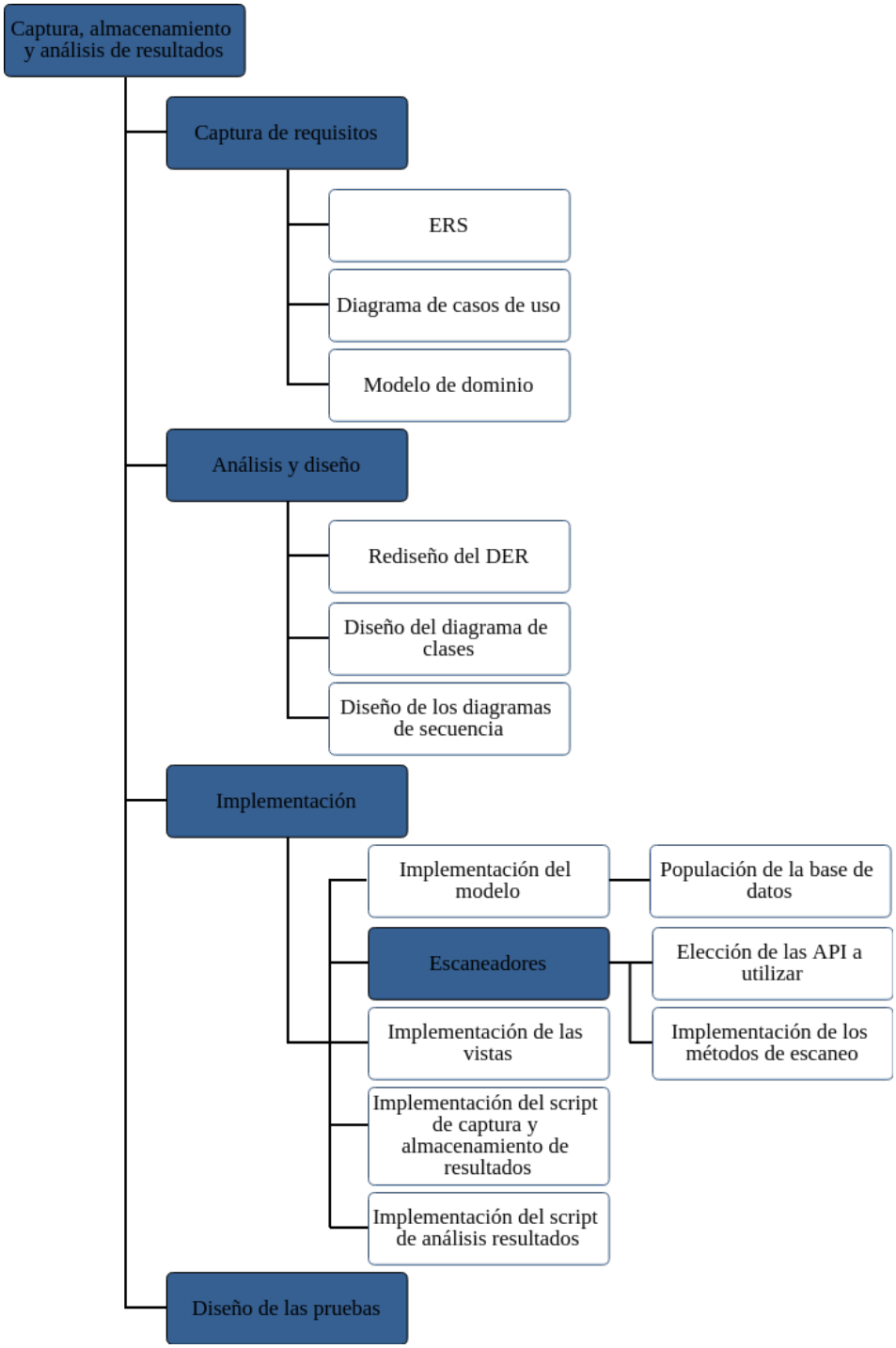


Figura 2.4: EDT correspondiente al módulo de Almacenamiento y análisis de resultados

2.5.2.1. Captura de requisitos

Este módulo abarca todas las tareas encargadas de recoger los requisitos que ha de satisfacer el backend, así como la elaboración de los diagramas destinados a describir éstos.

Especificación de Requisitos de Software
Paquete de trabajo: Captura de requisitos. **Duración estimada**: 8 horas.
Descripción: Realización de la Especificación de Requisitos de Software (ERS) de acuerdo al estándar de IEEE 830-1998[4]. **Entradas**: DOP. **Salidas/Entregables**: ERS. **Recursos necesarios**: Editor de textos, documentación en castellano para la correcta elaboración del ERS [Méndez, 2008].

Tabla 2.9: Especificación de Requisitos de Software del backend

Diagrama de casos de uso
Paquete de trabajo: Captura de requisitos. **Duración estimada**: 2 horas.
Descripción: Diseño del diagrama correspondiente para identificar y reflejar las acciones de los actores de la aplicación. **Entradas**: ERS. **Salidas/Entregables**: Diagrama de casos de uso. **Recursos necesarios**: Software para diseño de planificación de proyectos.

Tabla 2.10: Diagrama de casos de uso del backend

[4]El estándar IEEE 830-1998 fue reemplazado en 2011 por ISO/IEC/IEEE 29148:2011(E), más información en: `https://standards.ieee.org/findstds/standard/830-1998.html`

Modelo de dominio
Paquete de trabajo: Captura de requisitos. **Duración estimada**: 2 horas.
Descripción: Consiste en identificar y reflejar las entidades que conforman el sistema y las relaciones entre éstas, plasmándolo en el diagrama correspondiente. **Entradas**: Diagrama de casos de uso. **Salidas/Entregables**: Modelo de dominio. **Recursos necesarios**: Software para diseño de planificación de proyectos.

Tabla 2.11: Modelo de dominio del backend

2.5.2.2. Análisis y diseño

Recoge las tareas previas a la implementación del backend de la aplicación, partiendo de las especificaciones definidas y sirviendo para definir los métodos y propiedades que han de desarrollarse en los diferentes componentes del sistema.

Rediseño del DER
Paquete de trabajo: Análisis y diseño. **Duración estimada**: 8 horas.
Descripción: Definición de la nueva base de datos siguiendo un Diagrama de Entidad-Relación (DER), considerando que debe almacenar los resultados requeridos de forma estandarizada e independiente de la herramienta que se utilice, eliminando del modelo la parte inecesaria de versiones previas. **Entradas**: ERS, modelo de dominio. **Salidas/Entregables**: DER. **Recursos necesarios**: Software para diseño de planificación de proyectos y herramienta para el diseño de bases de datos.

Tabla 2.12: Rediseño del DER

Diseño del diagrama de clases
Paquete de trabajo: Análisis y diseño. **Duración estimada**: 2 horas.
Descripción: Realización del diagrama que represente e ilustre cada una de las clases con sus respectivos métodos y atributos, partiendo del modelo de dominio ya definido. **Entradas**: Modelo de dominio, DER. **Salidas/Entregables**: Diagrama de clases. **Recursos necesarios**: Software para diseño de planificación de proyectos.

Tabla 2.13: Diseño del diagrama de clases

Diseño de los diagramas de secuencia
Paquete de trabajo: Análisis y diseño. **Duración estimada**: 8 horas.
Descripción: Realización de los diagramas de secuencia para los métodos de las clases que así lo requieran. **Entradas**: Diagrama de clases. **Salidas/Entregables**: Diagramas de secuencia. **Recursos necesarios**: Software para diseño de planificación de proyectos.

Tabla 2.14: Diseño de los diagramas de secuencia

2.5.2.3. Implementación

Una vez finalizado el diseño, disponemos de la información necesaria para comenzar el desarrollo del backend de la aplicación, conformando las capa de persistencia de datos, así como la capa de negocio (denominadas también como vistas de *Django*).

Implementación del modelo
Paquete de trabajo: Implementación. **Duración estimada**: 4 horas.
Descripción: Creación y despliegue de la base de datos, de acuerdo al DER. **Entradas**: DER. **Salidas/Entregables**: Fichero *models* (véase: Arquitectura) y base de datos. **Recursos necesarios**: Herramienta para el diseño de bases de datos y entorno de desarrollo integrado.

Tabla 2.15: Implementación del modelo

Población de la base de datos
Paquete de trabajo: Implementación. **Duración estimada**: 4 horas.
Descripción: Implementación de un script que inserte los registros iniciales en las tablas de la base de datos que así lo requieran, así como información relevante almacenada desde versiones previas. Requiere analizar e identificar qué datos han de conservarse y, por ende, añadirse al script. **Entradas**: Fichero *models* del proyecto *Django* y base de datos. **Salidas/Entregables**: Script de población de la base de datos. **Recursos necesarios**: Entorno de desarrollo integrado.

Tabla 2.16: Población de la base de datos

2.5.2.4. Escaneadores

Probablemente uno de los módulos más extensos, en cuestión temporal: se basa en el estudio de las distintas alternativas entre las interfaces de programación de aplicaciones disponibles para la realización de escáneres sobre URL, a fin de seleccionar aquellas que mejor se ajusten al cometido

de la aplicación, dotándola de una mayor variedad que con la que se cuenta en la última versión disponible.

Elección de las API a utilizar
Paquete de trabajo: Escaneadores. **Duración estimada**: 20 horas.
Descripción: Análisis de las herramientas disponibles, así como de las que se utilizan en la última versión disponible del proyecto y elección de las que van a utilizarse en adelante. **Precedencias**: Implementación del modelo. **Salidas/Entregables**: Lista de las API a utilizar. **Recursos necesarios**: Internet, código y documentación de la versión previa .

Tabla 2.17: Elección de las API a utilizar

Implementación de los métodos de escaneo
Paquete de trabajo: Escaneadores. **Duración estimada**: 40 horas.
Descripción: Desarrollo de las funciones necesarias para implementar la totalidad de herramientas seleccionadas. Estudio de la información devuelta por éstas que queremos almacenar en la base de datos de la aplicación. **Entradas**: Lista de las API a utilizar. **Salidas/Entregables**: Funciones de los diferentes métodos de escaneo implementadas. **Recursos necesarios**: Entorno de desarrollo integrado e interfaces de programación de aplicaciones de las herramientas a utilizar.

Tabla 2.18: Implementación de los métodos de escaneo

Implementación de las vistas
Paquete de trabajo: Implementación. **Duración estimada**: 72 horas.
Descripción: Realización de los desarrollos asociados a la capa de negocio de la aplicación, de acuerdo al modelo definido. **Precedencias**: Implementación del modelo. **Salidas/Entregables**: Fichero *views* (véase: Arquitectura). **Recursos necesarios**: Entorno de desarrollo integrado.

Tabla 2.19: Implementación de las vistas

Implementación del script de captura y almacenamiento
Paquete de trabajo: Implementación. **Duración estimada**: 8 horas.
Descripción: Rediseño del script encargado del escaneo inicial de todas las páginas web almacenadas en la base de datos, así como de la actualización de los datos con un nuevo escaneo de las 50 páginas más consultadas cada mes. **Precedencias**: Implementación del modelo. **Salidas/Entregables**: Script *escanear_empresas.py*. **Recursos necesarios**: Entorno de desarrollo integrado.

Tabla 2.20: Implementación del script de almacenamiento

Implementación del script de análisis de resultados
Paquete de trabajo: Implementación. **Duración estimada**: 20 horas.
Descripción: Creación de gráficos de estadísticas en base a todos los sitios web analizados. El script se lanzará semanalmente, a fin de mantener los gráficos coherentes con lo almacenado en la base de datos. **Precedencias**: Implementación del modelo. **Salidas/Entregables**: Script *actualiza_graficos.py*. **Recursos necesarios**: Entorno de desarrollo integrado.

Tabla 2.21: Implementación del script de análisis de resultados

Diseño de las pruebas
Paquete de trabajo: Captura, almacenamiento y análisis de resultados. **Duración estimada**: 16 horas.
Descripción: Realización de las pruebas para proceder a sus comprobaciones una vez terminado el módulo de implementación. **Precedencias**: Implementación. **Salidas/Entregables**: Documentación de las pruebas. **Recursos necesarios**: Entorno de desarrollo integrado y editor de texto.

Tabla 2.22: Diseño de las pruebas

2.5.3. Interfaz de usuario web

Si el anterior módulo definía todas las fases relacionadas con el desarrollo del lado del servidor, este se centra en el frontend o entorno cliente de la aplicación. En este cabe destacar dos tareas principales: la implementación de las diferentes plantillas o *templates* de la página web y el rediseño de las hojas de estilos de ésta, a fin de dotar a la web de una interfaz gráfica con un diseño responsivo, que se adapte al dispositivo desde el que se acceda. Del mismo modo, se hará uso de HTML5 y CSS3.

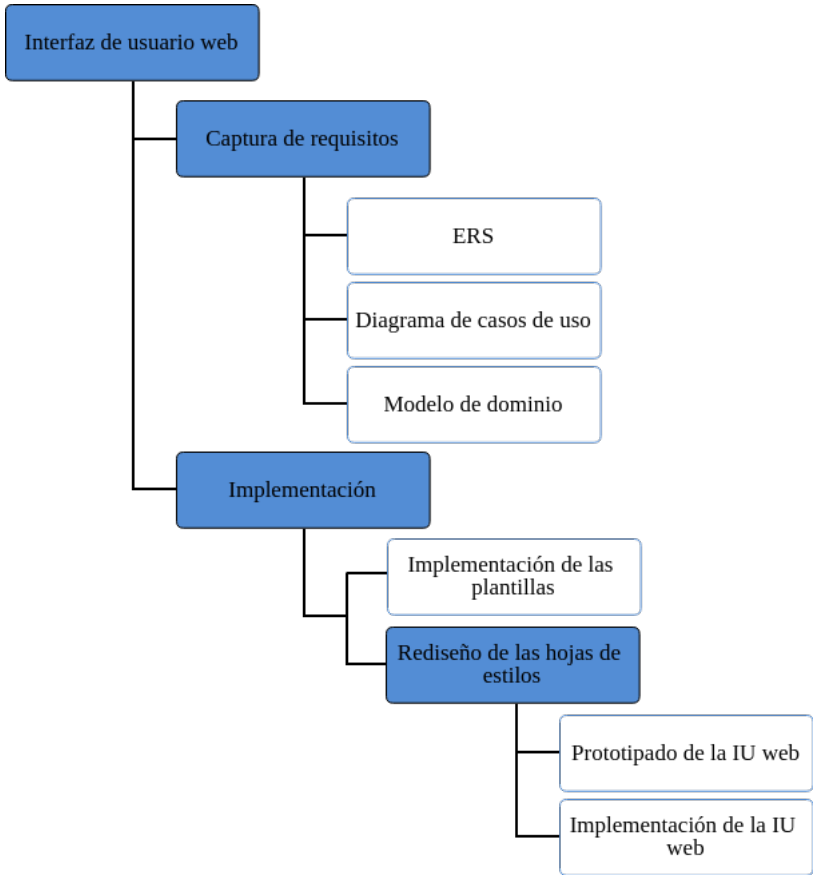


Figura 2.5: EDT correspondiente al módulo de Interfaz de usuario web

2.5.3.1. Captura de requisitos

Este módulo abarca todas las tareas encargadas de recoger los requisitos que ha de satisfacer el frontend, así como la elaboración de los diagramas destinados a describir éstos.

Especificación de Requisitos de Software
Paquete de trabajo: Captura de requisitos. **Duración estimada**: 4 horas.
Descripción: Realización de la Especificación de Requisitos de Software (ERS) de acuerdo al estándar de IEEE 830-1998. **Entradas**: DOP. **Salidas/Entregables**: ERS. **Recursos necesarios**: Editor de textos, documentación en castellano para la correcta elaboración del ERS.

Tabla 2.23: Especificación de Requisitos de Software del frontend

Diagrama de casos de uso
Paquete de trabajo: Captura de requisitos. **Duración estimada**: 4 horas.
Descripción: Diseño del diagrama correspondiente para identificar y reflejar las acciones de los actores de la aplicación. **Entradas**: ERS. **Salidas/Entregables**: Diagrama de casos de uso. **Recursos necesarios**: Software para diseño de planificación de proyectos.

Tabla 2.24: Diagrama de casos de uso del frontend

Modelo de dominio
Paquete de trabajo: Captura de requisitos. **Duración estimada**: 2 horas.
Descripción: Consiste en identificar y reflejar las entidades que conforman el sistema y las relaciones entre éstas, plasmándolo en el diagrama correspondiente. **Entradas**: Diagrama de casos de uso. **Salidas/Entregables**: Modelo de dominio. **Recursos necesarios**: Software para diseño de planificación de proyectos.

Tabla 2.25: Modelo de dominio del frontend

2.5.3.2. Implementación

Implementación de las plantillas
Paquete de trabajo: Implementación. **Duración estimada**: 8 horas.
Descripción: Implementación de los ficheros que contienen la información que se va a mostrar al usuario en las diferentes páginas que conforman la aplicación a desarrollar. Se va a hacer uso, en la medida de lo posible, de la semántica propia de la quinta revisión de HTML. **Entradas**: ERS, diagrama de casos de uso y modelo de dominio. **Salidas/Entregables**: Directorio *templates* y ficheros *html* (véase: Arquitectura). **Recursos necesarios**: Entorno de desarrollo integrado.

Tabla 2.26: Implementación de las plantillas

2.5.3.2.1 Rediseño de las hojas de estilos

El módulo abarca las tareas de prototipado e implementación de las plantillas que conforman el rediseño de la actual interfaz de usuario web de la aplicación. Este módulo es independiente de las plantillas o ficheros html, lo cual aísla los datos que se van a mostrar al usuario del formato con el que éstos se muestran.

Prototipado de la IU web
Paquete de trabajo: Rediseño de las hojas de estilos. **Duración estimada**: 4 horas.
Descripción: Rediseñar la interfaz de usuario web en partiendo del diseño actual y mediante prototipos a papel de las diferentes páginas. **Entradas**: ERS. **Salidas/Entregables**: Prototipos a papel. **Recursos necesarios**: Internet, código de la versión previa, papel y lápiz.

Tabla 2.27: Prototipado de la IU web

Implementación de la IU web
Paquete de trabajo: Rediseño de las hojas de estilos. **Duración estimada**: 36 horas (1 semana).
Descripción: Implementación del prototipo a papel en los ficheros necesarios, logrando un diseño responsivo y pudiendo realizarse éste con el apoyo de librerías externas (p. ej., *Bootstrap* [5]). Se va a hacer uso, en la medida de lo posible, de las nuevas funciones que otorga la tercera revisión de CSS. **Entradas**: Prototipos a papel. **Salidas/Entregables**: Directorio *static*, hoja de estilo en cascada y ficheros *JavaScript*. **Recursos necesarios**: Entorno de desarrollo integrado.

Tabla 2.28: Implementación de la IU web

2.5.4. Implantación

Detalla las tareas correspondientes al despliegue de la aplicación en el servidor y la configuración del acceso a ésta, así como la realización de las pruebas posteriores y la ejecución en segundo plano de los scripts de recopilación de información de páginas webs almacenadas por la base de datos del proyecto.

[5] *Documentación oficial de Bootstrap 4.1*: `https://getbootstrap.com/docs/4.1/`

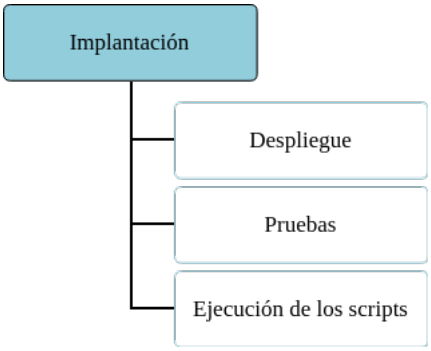


Figura 2.6: EDT correspondiente al módulo de Implantación

Despliegue
Paquete de trabajo: Implantación. **Duración estimada**: 2 horas.
Descripción: Instalación de la aplicación en el servidor y configuración del acceso a la misma desde el exterior. **Precedencias**: Almacenamiento y análisis de resultados e interfaz de usuario web. **Entradas**: Implementación completa del backend y frontend. **Recursos necesarios**: Acceso al servidor.

Tabla 2.29: Despliegue

Pruebas
Paquete de trabajo: Implantación. **Duración estimada**: 4 horas.
Descripción: Testeo del correcto funcionamiento de la aplicación. **Precedencias**: Despliegue. **Recursos necesarios**: Acceso al servidor y navegador.

Tabla 2.30: Pruebas

Ejecución de los scripts
Paquete de trabajo: Implantación. **Duración estimada**: 36 horas.
Descripción: Ejecución en segundo plano de los scripts desarrollados para el escaneo total de webs almacenadas y de generación de gráficos de resultados. **Precedencias**: Despliegue y pruebas. **Recursos necesarios**: Acceso al servidor.

Tabla 2.31: Ejecución de los scripts

2.5.5. Documentación

Este módulo describe las tareas con propósito de plasmar el esfuerzo dedicado a la redacción de la memoria del TFG. Del mismo modo, se van a documentar dos manuales, complementarios a la memoria y cuyo fin es, por un lado, ayudar en el proceso de instalación y configuración del servidor en el que se va a desplegar la aplicación y, por otra parte, explicar la herramienta para su uso online.

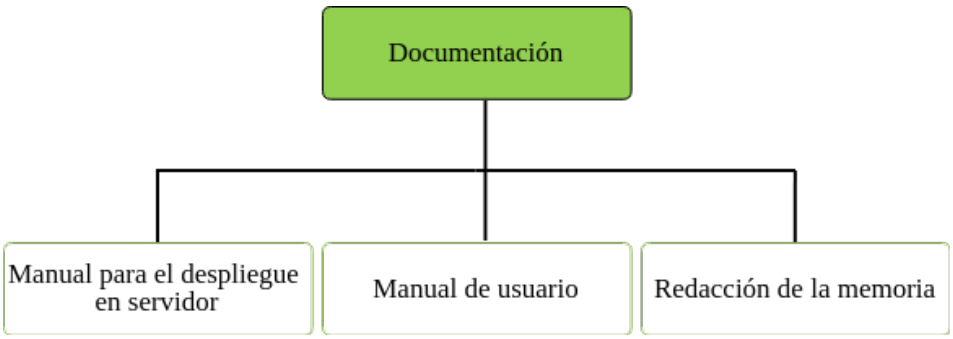


Figura 2.7: EDT correspondiente al módulo de Documentación

Manual para el despliegue en servidor
Paquete de trabajo: Documentación. **Duración estimada**: 2 horas.
Descripción: Redacción de un documento que sirva de referencia para la implatación de la aplicación en un sistema que cumpla con los requisitos necesarios. **Salidas/Entregables**: Manual para el despliegue en servidor. **Recursos necesarios**: Editor de textos.

Tabla 2.32: Manual para el despliegue en servidor

Manual de usuario
Paquete de trabajo: Documentación. **Duración estimada**: 4 horas.
Descripción: Redacción de un documento enfocado al perfil de un empresario que va a utilizar la aplicación, a fin de explicar las operaciones disponibles y de cara a facilitar su adaptación. **Salidas/Entregables**: Manual de usuario. **Recursos necesarios**: Editor de textos.

Tabla 2.33: Manual de usuario

Redacción de la memoria
Paquete de trabajo: Documentación. **Duración estimada**: 32 horas.
Descripción: Plasmar por escrito todas las tareas y acciones realizadas para la realización del TFG. **Precedencias**: Todas las tareas para la realización del TFG. **Salidas/Entregables**: Memoria. **Recursos necesarios**: Editor de textos.

Tabla 2.34: Redacción de la memoria

2.5.6. Resumen de tareas y precedencias

A modo de conclusión para la presente sección, se ha considerado útil reflejar la totalidad de las tareas a modo de tabla, en la que se reflejen los tiempos resultantes para cada módulo, así como las precedencias de cada tarea.

ID	***Nombre de la tarea***	***Precedencias***	***Duración estimada***
	EuskalCrawler III		490 horas
1	***Organización y aprendizaje***		138 horas
1.1	Planificación de las tareas		8 horas
1.2	Configuración de la máquina de desarrollo		2 horas
1.3	Creación y configuración del repositorio	1.2	2 horas
1.4	Aprendizaje de Django	1.2	72 horas
1.5	Análisis e interpretación del código actual	1.4	20 horas
1.6	Documento de Objetivos de Proyecto	1.1	34 horas
1.6.1	Elección de herramientas	1.1	2 horas
1.6.2	Diseño de diagramas	1.6.1	4 horas
1.6.3	Redacción del DOP	1.6.2	28 horas

Tabla 2.35: Tareas y precedencias del módulo de Organización y aprendizaje

ID	*Nombre de la tarea*	*Precedencias*	*Duración estimada*
2	***Captura, almacenamiento y análisis de resultados***	1.6	214 horas
2.1	Captura de requisitos	1.6	12 horas
2.1.1	Especificación de Requisitos de Software	1.6	8 horas
2.1.2	Diagrama de casos de uso	2.1.1	2 horas
2.1.3	Modelo de dominio	2.1.2	2 horas
2.2	Análisis y diseño	2.1	18 horas
2.2.1	Rediseño del Diagrama de Entidad-Relación	2.1.3	8 horas
2.2.2	Diseño del diagrama de clases	2.1	2 horas
2.2.3	Diseño de los diagramas de secuencia	2.2.2	8 horas
2.3	Implementación	2.2	168 horas
2.3.1	Implementación del modelo	2.2.1	4 horas
2.3.1.1	Población de la base de datos	2.3.1	4 horas
2.3.2	Escaneadores	2.3.1	60 horas
2.3.2.1	Elección de las API a utilizar	2.3.1	20 horas
2.3.2.2	Implementación de los métodos de escaneo	2.3.2.1	40 horas
2.3.3	Implementación de las vistas	2.3.1	72 horas
2.3.4	Implementación del script de almacenamieto	2.3.1	8 horas
2.3.5	Implementación del script de análisis de resultados	2.3.1	20 horas
2.4	Diseño de las pruebas	2.3	16 horas

Tabla 2.36: Tareas y precedencias del módulo de Captura, almacenamiento y análisis de resultados

ID	*Nombre de la tarea*	*Precedencias*	*Duración estimada*
3	***Interfaz de usuario web***	1.6	58 horas
3.1	Captura de requisitos	1.6	10 horas
3.1.1	Especificación de Requisitos de Software	1.6	4 horas
3.1.2	Diseño de los casos de uso	3.1.1	4 horas
3.1.3	Diseño del modelo de dominio	3.1.2	2 horas
3.2	Implementación	3.1	48 horas
3.2.1	Implementación de las plantillas	3.1	8 horas
3.2.2	Rediseño de las hojas de estilos	3.1	40 horas
3.2.2.1	Prototipado de la IU web	3.1	4 horas
3.2.2.2	Implementación de la IU web	3.2.2.1	36 horas
4	***Implantación***	2, 3	42 horas
4.1	Despliegue	2, 3	2 horas
4.2	Pruebas	4.1	4 horas
4.3	Ejecución de los scripts	4.2	36 horas
5	***Documentación***		38 horas
5.1	Manual para el despliegue en el servidor	4	2 horas
5.2	Manual de usuario	4	4 horas
5.3	Redacción de la memoria	1, 2, 3, 4, 5.1, 5.2	32 horas

Tabla 2.37: Tareas y precedencias de los módulos Interfaz de usuario web, Implantación y Documentación

2.6. Gestión de riesgos

Ningún proyecto está exento de posibles contratiempos e imprevistos, si bien anticiparse a ellos, previo análisis, puede ayudar a minimizar sus consecuencias.

A continuación, se enumeran los riesgos tenidos en cuenta, detallando para cada uno de los mismos los planes de prevención y de contingencia, a fin de establecer cómo actuar para evitar su aparición y reducir su impacto, respectivamente. Del mismo modo, se establecerá en cada caso la probabilidad de aparición e impacto, en cuanto a pérdida de tiempo de trabajo se

refiere, de acuerdo a la jornada establecida (véase: Método de trabajo).

Se ha establecido una medida de impacto de acuerdo a la pérdida de horas de trabajo semanales, tal que: se considerará el impacto *muy bajo* si la pérdida horaria no supera a 1 hora, *bajo* si ésta oscila entre 1 y 2 horas, *medio* para valores de 2 a 3 horas, *alto* para 4 horas (equivalente a un día de trabajo) y *crítico* para pérdidas mayores a un día de trabajo. Se ordenarán los riesgos en base a este criterio, de mayor a menor impacto.

2.6.1. Posibilidad de contratación

Partiendo al inicio del proyecto de una situación laboral de contrato en prácticas a media jornada, cabe la posibilidad de incorporación definitiva a la empresa, mediante un contrato a jornada completa, con la pérdida de horas disponibles para la dedicación al avance en el TFG que ello conllevaría.

Plan de prevención

- Cumplimiento de la planificación temporal.
- Establecer una holgura de al menos dos semanas desde la fecha de finalización del proyecto hasta la fecha límite de la entrega, a fin de contemplar posibles ajustes en la planificación temporal.

Plan de contingencia

- Modificación de la jornada laboral del proyecto y su distribución diaria, disminuyendo o eliminando las horas de trabajo entre semana en pro de un aumento en la dedicación al proyecto a lo largo del fin de semana.
- Modificación del cómputo semanal de horas dedicadas al desarrollo del proyecto.

Probabilidad

Alta: 85 %

Impacto

0,85 x 5 días = 17 horas. Impacto crítico.

2.6.2. Parálisis por análisis

Es el término, relacionado con la procrastinación y el exceso de perfeccionismo, por el que se conocen aquellas situaciones excesivamente reflexivas que dificultan la toma de decisiones. Su aparición se encuentra muy ligada a los proyectos de desarrollo de software (llegando incluso a considerarse un *antipatrón de diseño* [Brown *et al.*, 1998]), en los momentos en que se detiene el avance por culpa de un análisis tan exhaustivo que imposibilita la elección de entre una cantidad ingente de información, debido al miedo a escoger la opción incorrecta.

Plan de prevención

- Definir con exactitud las tareas en la elaboración del alcance.
- Correcta planificación temporal inicial.

Plan de contingencia

- Dividir un problema grande en problemas más pequeños.
- Búsqueda en foros relacionados con el tema bloqueante.
- Consulta al director del proyecto en busca de consejo.
- Pasar temporalmente a otra actividad.

Probabilidad

Alta: 75 %

Impacto

0,70 x 1 día = 3 horas. Impacto medio.

2.6.3. Pérdida de la base de datos

Principal riesgo a tener en cuenta en el proyecto, dado el extenso nivel de información relativa a sitios webs de empresas almacenada aún desde versiones previas del proyecto, y consistente en pérdida de información de páginas web de empresas vascas y/o resultados de los escaneos a éstas.

Plan de prevención

- Generación de un script de población con la información inicial, de cara a restaurar la información almacenada automáticamente en caso de pérdida parcial.
- Realización de una copia de seguridad completa diaria de la base de datos.

Plan de contingencia

- Recuperación de información de la base de datos por medio de un script, en caso de pérdida de parcial.
- Restablecimiento de la última copia de seguridad creada, en caso de pérdida total.

Probabilidad

Baja: 35 %

Impacto

0,35 x 1 día = 1,4 horas. Impacto bajo (en caso de no haber implementado el plan de prevención, impacto crítico).

2.6.4. Enfermedad o lesión

Consiste en la indisponibilidad por una causa médica, bien derivada del trabajo o por causas ajenas al mismo.

Plan de prevención

- Derivadas del trabajo:
  - Mantener una postura correcta a la hora de trabajar.
  - Mantener la distancia correcta frente al monitor.
  - Realizar *coffee breaks* de 5 a 10 minutos cada 2 horas de trabajo.
- No derivadas:
  - Dormir las horas recomendadas por los expertos en salud.
  - Realizar actividad deportiva, evitando actividades de riesgo.

Plan de contingencia

- Acudir al médico para diagnosticar el alcance de la lesión o enfermedad.
- En caso de que el diagnóstico sea favorable, se continuará con el trabajo, tratando de descansar más a menudo.
- En caso de que el diagnóstico sea desfavorable, se aplazará el trabajo el tiempo necesario para recuperarse y poder continuar.

Probabilidad

- Diagnóstico favorable. Baja: 15 %.
- Diagnóstico desfavorable. Muy baja: 5 %.

Impacto

- Diagnóstico favorable. 0,15 x 2 días = 1,2 horas. Impacto bajo.
- Diagnóstico desfavorable. 0,05 x 5 días = 1 hora. Impacto bajo.

2.6.5. Pérdida del código

Consiste en la pérdida de horas de trabajo — en cuestión de código del que no han sido guardados los cambios —, debido a un fallo en el suministro eléctrico, despiste o deterioro del medio en el que se encontraban almacenados los ficheros.

Plan de prevención

- Utilización de un repositorio en línea al cual subir los cambios realizados.
- Realizar una subida de forma regular y diaria (al menos más de una vez por jornada), aún si hay desarrollos en curso y/o con errores.
- Descarga de los cambios locales en el servidor, previa subida al repositorio.

Plan de contingencia

- Recuperación de la última versión almacenada en el repositorio.
- En caso de problemas con el repositorio, descarga a local de la última versión disponible en el servidor o viceversa.

Probabilidad

Baja: 15 %

Impacto

0,15 x 1 día = 0,6 horas. Impacto muy bajo.

2.6.6. Deterioro del equipo informático

Engloba los posibles daños, debidos a acciones relacionadas con el trabajo, transporte de material o agentes externos, en el equipo utilizado para la elaboración el proyecto.

Plan de prevención

- Mantenimiento del equipo en unas condiciones ambientales adecuadas.
- Evitar la sobrecarga del equipo debido a su uso para fines diferentes a la propia realización del proyecto.
- Revisión periódica del estado del equipo.

Plan de contingencia

- Para fallos derivados de componentes hardware, comprar y reemplazar el componente en cuestión a mayor brevedad posible, preferiblemente el mismo día de la avería.
- En caso de avería grave, se procederá a utilizar un equipo disponible en el aula de ordenadores de la propia universidad hasta la reparación del equipo de trabajo.

Probabilidad

Muy baja: 10 %

Impacto

0,10 x 1 día = 0,4 horas. Impacto muy bajo.

2.6.7. Pérdida de la conexión a Internet

Como su propio nombre indica, refleja los contratiempos derivados de perder la conexión mientras se está trabajando, tales como la imposibilidad de comunicarse con el servidor o con el repositorio de versiones del proyecto, así como la falta de un medio en el que buscar información y/o consultar documentación.

Plan de prevención

- Contratación de una conexión a Internet con velocidad suficiente.
- Utilización de conexión a Internet por cable de forma preferente frente a una conexión vía WiFi.
- Evitar descargas de ficheros y ejecución de servicios que consuman recursos de Internet ajenos al proyecto en el horario establecido de trabajo.

Plan de contingencia

- Utilizar el móvil para conectarse a Internet desde el portátil (conocido como *tethering*).
- Trasladarse a la universidad o biblioteca, a fin de continuar con el desarrollo en un aula de trabajo habilitada para estudiantes.

Probabilidad

Muy baja: 5 %

Impacto

0,05 x 1 día = 0,4 horas. Impacto muy bajo.

2.7. Planificación temporal

Conociendo los módulos — y respectivo desglose — que componen el proyecto y habiendo establecido el *deadline* a 29 de junio, así como analizado los posibles riesgos que pudieran entorpecer su desarrollo y definido las medidas pertinentes en cada caso, es el momento de establecer la distribución en el tiempo de todas las tareas definidas. Como es habitual, se va a representar por medio de un diagrama de Gantt, cuyo resultado se muestra en la Figuras 2.8, 2.9, 2.10 y 2.11.

Se recuerda que el cómputo horario semanal es de 36 horas (véase: Método de trabajo), así como que el cálculo de horas totales, según la duración estimada de las tareas asciende a un total de 490 horas (véase: Alcance).

Cabe mencionar que debido a la sinergia existente entre ciertas tareas, éstas pueden realizarse de forma conjunta (tareas Despliegue del módulo Implantación y tarea Manual para el despliegue en servidor del módulo Documentación); unido ello a que el proyecto es realizado por una única persona, para estos casos las tareas reflejadas en el diagrama verán multiplicadas sus horas por el número de tareas realizadas de forma pareja, por lo que dos tareas de dos horas tardarían igualmente cuatro horas en realizarse, si estas se realizan al mismo tiempo.

Esta planificación es una estimación en base al planteamiento inicial y, como tal, es susceptible a cambios a lo largo del ciclo de vida del proyecto.

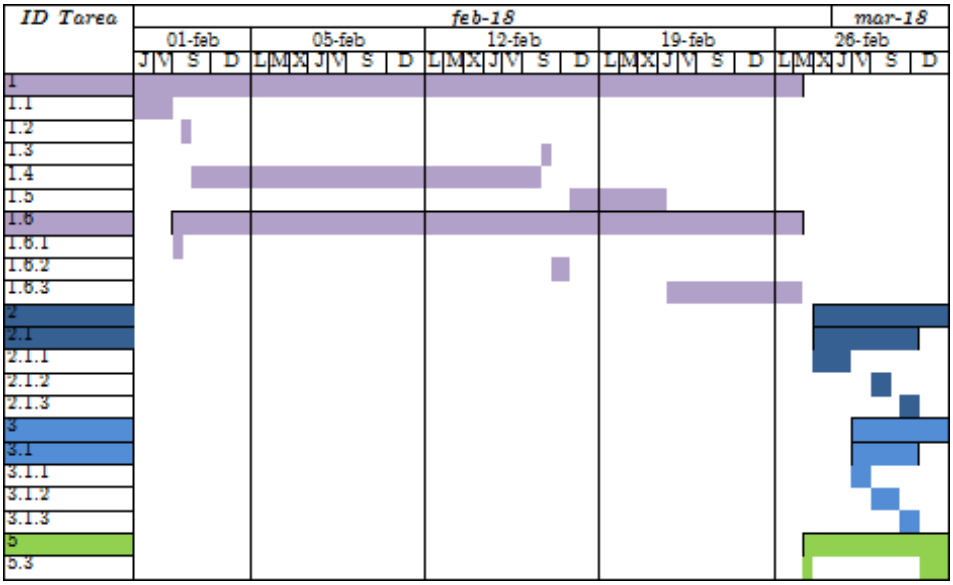


Figura 2.8: Diagrama de Gantt del mes de febrero

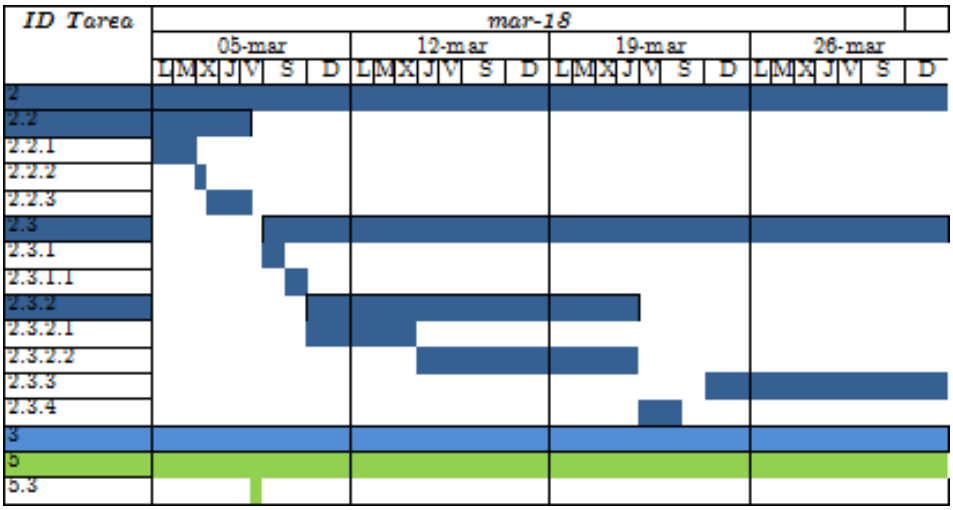


Figura 2.9: Diagrama de Gantt del mes de marzo

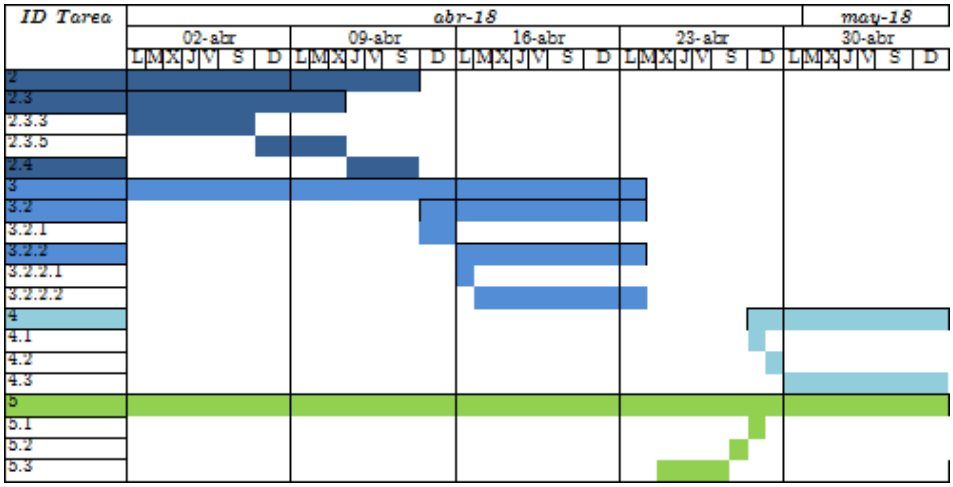


Figura 2.10: Diagrama de Gantt del mes de abril

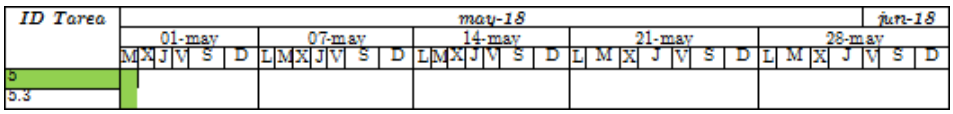


Figura 2.11: Diagrama de Gantt del mes de mayo

Como se puede apreciar en la tabla correspondiente al mes de mayo, la previsión es finalizar el proyecto al comienzo del mes, lo cual encaja con el objetivo anteriormente planteado de entrega entre los días 28 y 29 de junio (véase: Objetivos), con una holgura superior a un mes, a fin de disponer de tiempo suficiente en caso de surgir algún contratiempo a lo largo del desarrollo del mismo.

2.8. Evaluación económica

Cerrando el DOP, se va a realizar un presupuesto con el que calcular el coste de desarrollo de todas las etapas del proyecto y del que valerse para el supuesto de que la aplicación llegue a comercializarse, si bien esta no es la idea actual del presente TFG.

2.8.1. Mano de obra

Como el trabajo realizado se enmarca en la categoría de Analista-Programador y Diseñador pagina web, se ha empleado para el cálculo el sueldo mínimo para esta categoría del Convenio colectivo estatal de empresas de consultoría y estudios de mercados y de la opinión pública del presente año, en base a las tablas publicadas para el convenio de 2018 en el BOE a fecha de 6 de marzo de 2018. En dicho documento, se fija un sueldo base (plus de convenio incluido) de 1481,22€ mensuales (Anexo I, Tablas salariales [Ministerio de Empleo y Seguridad Social, 2018]).

Dado que la jornada laboral establecida para la elaboración del proyecto, en cuanto a horas semanales y distribución de éstas se refiere, no se ajusta a lo establecido para un contrato a jornada completa, hay que calcular en primer lugar el salario por hora de trabajo, en base a:

$$Sueldo_{hora} = \frac{Sueldo_{mes}}{Horas\ de\ trabajo_{semana} \cdot Semanas_{mes}} \tag{2.1}$$

De acuerdo al cómputo horario semanal (véase: Método de trabajo), fijado en treinta y seis horas, el resultado final es:

$$Sueldo_{hora} = \frac{1481.22€}{36 \cdot 4} = \frac{1481.22€}{144} = 10.29€ \tag{2.2}$$

Es ahora, y con esta cifra como referencia, cuando podemos proceder al cálculo de la mano de obra derivada del total de horas de trabajo del proyecto:

$$Mano\,de\,obra = Horas\,de\,trabajo_{total} \cdot Sueldo_{hora} = 490 \cdot 10.29 € = 5042.10 €$$ (2.3)

Concluyendo en un gasto económico, en lo que a mano de obra se refiere, asciende a 5042.10 €.

2.8.2. Gasto en equipo informático

El siguiente aspecto a tratar es el gasto económico derivado de la amortización del equipo informático utilizado para la elaboración del proyecto.

2.8.2.1. Ordenador portátil

Para el desarrollo del proyecto, se cuenta con un ordenador portátil marca *MSI* y modelo *GE60 2PC Apache*, con un precio de 995 € y cuya vida útil estimada es de 6 años.

El cálculo del gasto que supone se realiza en base a su amortización mensual, que se realiza de acuerdo a la fórmula siguiente:

$$Amortización_{mensual} = \frac{Precio}{Vida\ útil\ estimada} = \frac{995}{\sim 72\,\text{meses}} = 13.82 €$$ (2.4)

Obteniendo la amortización mensual y suponiendo una vida útil de 72 meses, podemos calcular la amortización acumulada durante el ciclo de vida del proyecto, de acuerdo a los cálculos vistos previamente:

$$Amortización_{hora} = \frac{13.82 €}{144} = 0.10 €$$ (2.5)

$$Amortización_{total} = 490 \cdot 0.10 € = 49 €$$ (2.6)

Obtenemos que el gasto derivado de la amortización del portátil es de un total de 49 €.

2.8.2.2. Servidor

Se va a hacer uso de la misma máquina que albergaba el proyecto en su anterior etapa, el cual es un servidor *Microsoft Azure* cuyo coste de mantenimiento anual es de 100 € [6].

Para este caso, y al entenderse que el pago de dicho servicio ya se ha realizado, se incluye la totalidad de dicho importe como gastos de equipo informático del proyecto.

2.8.3. Gasto en software

La totalidad del proyecto, tanto en lo que se refiere a desarrollo como en la parte referente a documentación, se va a realizar utilizando un sistema operativo de la familia *Linux*, evitando el uso de software propietario. No obstante, las herramientas a utilizar que requieran de licencias no implicarán la compra de éstas, gracias a las licencias para uso educativo otorgadas por la universidad a sus estudiantes.

2.8.4. Gastos indirectos

No podemos dejar de lado aquellos gastos que, si bien no entran dentro de la propia actividad productiva del proyecto, suponen un coste que es necesario reflejar en la evaluación económica.

2.8.4.1. Luz

Para concretar el gasto de luz derivado del proyecto, es necesario conocer la potencia en kilovatios del equipo informático utilizado. En este caso, únicamente contamos con el portátil anteriormente descrito, ya que el servidor es un servicio remoto y, como tal, no se ubica en la zona de trabajo ni hace uso de su red eléctrica.

Según las especificaciones del *MSI GE60 2PC Apache*[7], la potencia de su adaptador es de 120 W = 0.120 kW.

[6] De acuerdo a la documentación relativa a la segunda fase del proyecto [Gallego, 2018]

[7] Página oficial de especificaciones del producto: `https://es.msi.com/Laptop/GE60-2PC-Apache/Specification`

Por otra parte, el precio del kilovatio hora (kWh) de la compañía contratada en el lugar de trabajo es de 0.121 €[8].

Por lo tanto, en base a la estimación horaria de la planificación temporal y a que el uso del equipo informático es continuo e imprescindible en la elaboración del proyecto, podemos obtener el gasto energético de acuerdo a:

$$Gasto\ luz_{total} = 490 \cdot 0.121\,€ \cdot 0.120\,\text{kW} = 7.12\,€ \tag{2.7}$$

Contamos con un gasto de luz, fruto del uso del portátil, de un coste total por valor de 7.12 €.

2.8.4.2. Internet

Para la elaboración del proyecto, la conexión ininterrumpida a Internet resulta imprescindible. Adicionalmente, hay que contar con la tarifa de datos del teléfono móvil, la cual se ha establecido que se utilizará en caso de pérdida de la conexión (véase: Gestión de riesgos).

- **Cuota Internet**: 55 € mensuales, para los cuatro meses de duración del proyecto obtenemos un total de 220 €.
- **Tarifa de datos móvil**: 19.90 € mensuales, un total de 79.60 €.

[8] Datos extraídos de: `https://tarifasgasluz.com/faq/precio-kwh`

2.8.5. Gastos totales

En la Tabla 2.38 se resumen los gastos originados de la elaboración del proyecto y se refleja la suma total de éstos.

Concepto	***Gasto (en €)***
Mano de obra	5042.10 €
Gasto en equipo informático	149.00 €
Gasto en software	0 €
Gastos indirectos	306.72 €
Total	5497.82 €

Tabla 2.38: Resumen de gastos y total

En resumidas cuentas y a modo de cierre, el coste del proyecto en base a una planificación temporal por módulos y tareas, cuyas duraciones son fruto de una estimación inicial, se fija en un total de 5497.82 €.

3. Captura de requisitos

En el presente capítulo expone la Especificación de Requisitos de Software (ERS) del proyecto, siguiendo el antiguo estándar de IEEE 830-1998 para su definición y especificando los requisitos que han de ser satisfechos por el sistema.

Se incluyen, de forma complementaria, los diagramas correspondientes a los casos de uso y modelo de dominio resultantes de dicha especificación.

Definiciones, acrónimos y abreviaturas

De aquí en adelante, se referirá a Requisitos Funcionales y Requisitos de Interfaces Externos como RF y RIE respectivamente, seguidos ambos de su correspondiente numeración.

3.1. Descripción general

A continuación, se describen los factores que afectan al producto y a sus requisitos, en lo que respecta al contexto de los mismos.

3.1.1. Perspectiva del producto

Se pretende realizar un rediseño e implementación de una aplicación web, orientada a un uso empresarial, de una herramienta de escaneo de sitios web de la que se obtiene una lista de información relevante y posibles vulnerabilidades.

El aspecto novedoso del proyecto reside en la utilización de varias interfaces de programación de aplicaciones de terceros, cuyo uso de forma paralela y estandarizada permite recoger y almacenar una gran cantidad de información acerca de las tecnologías utilizadas en la elaboración del sitio web analizado, en base a la cual devolver resultados veraces.

3.1.2. Funciones del producto

En lo que respecta a la aplicación web, los usuarios finales podrán añadir sitios web de empresas de las cuales sean propietarios (para lo cual se hará uso de un sistema de verificación) y acceder a la herramienta de escaneo, a fin de realizar un análisis de los mismos. Tras ello, podrán tener acceso a los resultados de forma inmediata — mostrándose dichos datos con la mayor simplicidad posible — o realizar un nuevo escaneo siempre que se considere pertinente.

Mediante lo anterior mencionado, se mantendrá una base de datos que irá incrementando su información de forma colaborativa, fruto de los escaneos realizados por los usuarios, de la cual se realizará un escaneo general de todos los sitios web almacenados al inicio de la presente fasese y de la que se obtendrán, periodicamente, datos estadísticos del desarrollo de los sitios web de empresas vascas. Estos resultados serán también visibles por los usuarios que accedan a la aplicación.

3.2. Requisitos específicos

A continuación se procede a detallar la serie de requisitos que el sistema ha de satisfacer.

3.2.1. Requisitos funcionales

RF1. Alta de usuario

El sistema permitirá a los usuarios no registrados darse de alta en el mismo. Para ello, el usuario deberá acceder a la página de registro e introducir los siguientes datos:

- **Nombre de usuario**: obligatorio y que deberá ser único en el sistema.
- **Email**: obligatorio y que deberá ser único en el sistema.
- **Contraseña**: obligatoria y que deberá ser confirmada por el propio usuario.

En el momento del alta, el sistema asignará al usuario un código de

activación pseudo-aleatorio[1] y único y enviará un correo a la dirección facilitada por el propio usuario, solicitando la confirmación del registro en un periodo de dos días y mediante un enlace de activación que haga uso del código generado.

RF2. Acceso al Sistema

Para acceder al sistema el usuario deberá validarse como tal en el mismo, introduciendo en la ventana de inicio los siguientes datos:

- **Nombre de usuario** o **correo electrónico**. Podrá validarse con ambos de manera indiferente.
- **Contraseña**. Introducirá la contraseña una sola vez, sin ser necesaria la confirmación de esta.

Una vez introducidos los datos se dará la opción al usuario de confirmarlos para intentar el acceso al sistema. En caso de error en la autenticación será retornado a la misma página con un mensaje que especifique claramente la razón por la que esta ha fallado.

RF3. Desconexión de usuario

El usuario podrá desconectarse del sistema seleccionando la opción mostrada a lo largo de todas las páginas de la aplicación, situada siempre en la parte superior derecha. Una vez desconectado será automáticamente reenviado a la página de inicio.

RF4. Consulta de sitios web

El sistema ofrecerá a los usuarios autenticados la posibilidad de realizar una consulta de todos los sitios web dados de alta por él mismo, si los hubiere, desde la ventana que los gestione.

RF5. Alta de sitio web

El sistema ofrecerá a los usuarios autenticados la posibilidad de dar de alta sitios web relativos a empresas de su propiedad, a fin de realizar un posterior escaneo de los mismos. Estos sitios web serán dados de alta

[1] un código pseudo-aleatorio es aquel generado en un proceso que parece producir códigos al azar, pero que realmente es generado por un algoritmo completamente determinista, en el que las mismas condiciones iniciales producen siempre el mismo resultado

desde la ventana que los gestione, en la cual el usuario deberá introducir el siguiente dato:

- **URL**: obligatoria en la operación de alta de un sitio web, única para el usuario que la da de alta (si bien varios usuarios podrán tener asociado un mismo sitio web) y con un máximo de 200 caracteres.

Una vez introducida la URL y comprobado que es correcta se realizará una recarga de la misma página con el sitio web almacenado. En caso de producirse un error, el sistema advertirá al usuario con un mensaje descriptivo de la situación.

RF6. Verificación de sitio web

El sistema ofrecerá a los usuarios autenticados y con algún sitio web de empresa registrado la posibilidad de verificar la autoría de éste. Para ello, se solicitará al usuario la creación de un contenido específico en el sitio web, basado en un fichero cuyo título y cuerpo sean ambos un número pseudo-aleatorio de 16 dígitos, que se facilitará al usuario en la ventana que lo gestione. El propio sistema se encargará de comprobar, de forma automática, que esta acción ha sido realizada por el usuario.

RF7. Escaneo de sitio web

El sistema dará la posibilidad a los usuarios autenticados de realizar un escaneo de sus sitios web de empresas registrados y verificados desde la ventana que los gestione.

RF8. Consulta de resultados de sitio web

El sistema dará la posibilidad a los usuarios autenticados de consultar los resultados de los escaneos realizados a sus sitios web de empresas desde la ventana que los gestione.

RF9. Consulta de datos de interés

El sistema dará la posibilidad consultar datos estadísticos, reflejados en forma de gráficos, de los resultados globales y almacenados en la base de datos, desde la ventana que los gestione. Estos datos se actualizarán de forma automática y con una periodicidad semanal.

RF10. Escaneo automático

El sistema deberá realizar, de forma automática y con una periodicidad mensual, un escaneo de los sitios web de empresas registradas, abarcando la totalidad de empresas la primera vez y centrándose en las cincuenta más visitadas, si las hubiere, en los posteriores escaneos.

3.2.2. Requisitos de interfaces externos

RIE1. Compatibilidad con navegadores

En lo que respecta a la interfaz de usuario web, el sistema deberá funcionar con una operatividad total en los siguientes navegadores web:

- **Google Chrome**. A partir de la versión 45.0.
- **Mozilla Firefox**. A partir de la versión 38.0.
- **Internet Explorer / Microsoft Edge**. A partir de la versión 10.0.

En todos ellos deberán realizarse las pruebas pertinentes para comprobar el correcto funcionamiento. No obstante, pueden presentarse algunos cambios de interfaz entre los navegadores enumerados.

RIE2. Diseño responsivo

A fin de satisfacer también las necesidades de los usuarios que accedan a la interfaz de usuario web mediante sus dispositivos móviles, el sistema deberá estar dotado de un diseño responsivo, que deberá funcionar con una operatividad total en los navegadores por defecto de los siguientes sistemas operativos móviles:

- **Android**. A partir de la versión 5.0.
- **iOS**. A partir de la versión 9.0.
- **Windows Mobile**. A partir de la versión 10.0.

3.3. Jerarquía de actores

Se han definido un *usuario autenticado* y otro *anónimo*, tal como se aprecia en la Figura 3.1.

Figura 3.1: Jerarquía de actores

Se considera apropiada además la inclusión de un tercer actor, al que se referenciará con el nombre de *administrador*, el cual podrá ejercer control sobre los casos de uso que el sistema ha de realizar de forma automática (p. ej., será el encargado de configurar la periodicidad con la que ha de ejecutarse el script de escaneo de los sitios web de empresas registradas).

3.4. Casos de uso

En sincronía con los requisitos funcionales definidos y los actores identificados en el sistema, el diagrama resultante presenta el aspecto que expone la Figura 3.2.

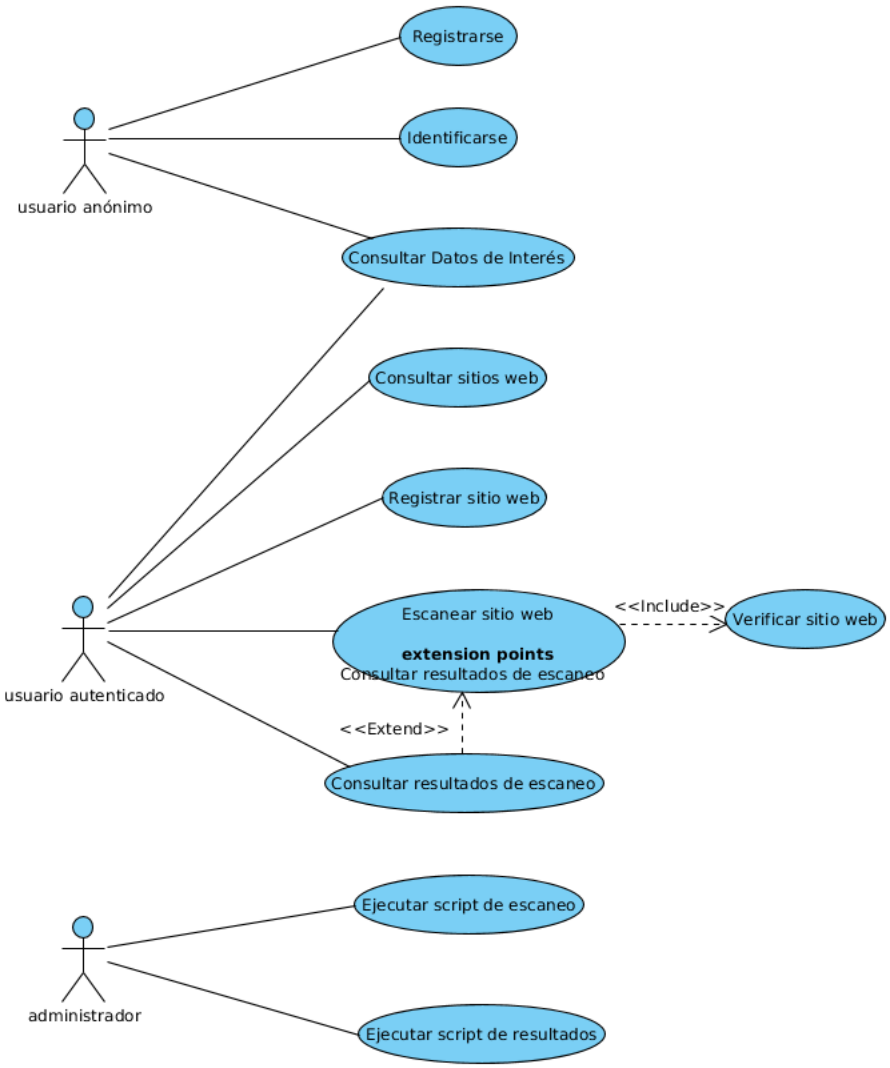


Figura 3.2: Casos de uso

3.5. Modelo de dominio

Previo al diseño del modelo y de su correspondiente Diagrama de Entidad-Relación, se va a representar un modelo de dominio, del cual partir como base, que contenga la información imprescindible a conservar de las versiones previas y que conforma la estructura básica del proyecto. En base a la ERS, obtenemos el diagrama de la Figura 3.3.

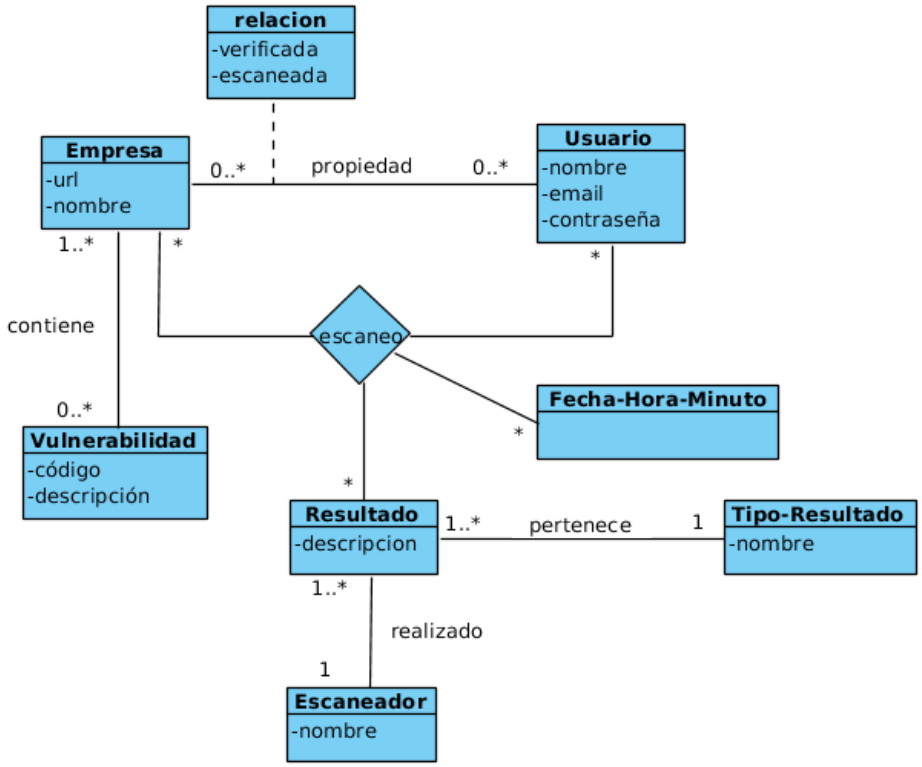


Figura 3.3: Modelo de dominio

3.5.1. Entidades

Usuario

De acuerdo a los requisitos, define a cada empresario registrado en la aplicación, del que se almacenan un nombre identificativo en el sistema, email y contraseña.

Empresa

Hace referencia a la entidad encargada de recoger información básica de una empresa y, por ende, del sitio web de ésta, con información como la URL o el nombre de la empresa a la que hace referencia.

Escaneador

Identifica cada herramienta de escaneo de la que hace uso la aplicación.

Resultado

Registra cada unidad de información relevante fruto del escaneo de un sitio web.

Tipo resultado

Guarda el nombre de los diferentes tipos de resultados que pueden obtenerse.

Vulnerabilidad

Contiene el código identificativo único de la CVE en cuestión, así como una descripción de la misma.

Fecha-Hora-Minuto

Permite almacenar la fecha, con hora y minuto, de diferentes acciones ejecutadas por la aplicación. Es de utilidad para almacenar un histórico de datos del mismo tipo.

3.5.2. Relaciones

Propiedad

Establece un vínculo binario entre una empresa y un usuario, que indica la propiedad de la primera por parte del segundo. Contiene como atributos los valores booleanos que hacen referencia a si la empresa está verificada y/o escaneada para dicho propietario.

Escaneo

Es una relación múltiple entre el usuario, la empresa, la serie de resultados obtenidos tras el escaneo y el instante en que se realiza.

Contiene

Relaciona una vulnerabilidad con la empresa (y por definición, el sitio web) que la contiene.

Realizado

Describe qué escaneador ha sido el autor de un resultado concreto.

Pertenece

Relación binaria que define, por cada resultado, el tipo al que éste pertenece.

4. Análisis y diseño

Se procederán a detallar en el presente capítulo tareas previas a la fase de desarrollo del proyecto, tales como el Diagrama de Entidad-Relación (DER) resultante del rediseño de la base de datos, así como el diagrama de clases de la aplicación.

Del mismo modo, se va a explicar — y representar, para los casos que se considere conveniente — el diseño de los principales métodos del sistema, que servirán para implementar posteriormente el proceso de escaneo individual y global, así como el análisis y presentación de resultados obtenidos.

4.1. Diagrama de base de datos

Al tratarse de una base de datos que sigue el modelo relacional, el presentado en la Figura 4.1 es en concreto un Diagrama de Entidad-Relación (DER). Se han omitido las tablas autogeneradas por *Django* al crear un nuevo proyecto, a excepción de *auth_user*, que contiene la información referente al usuario.

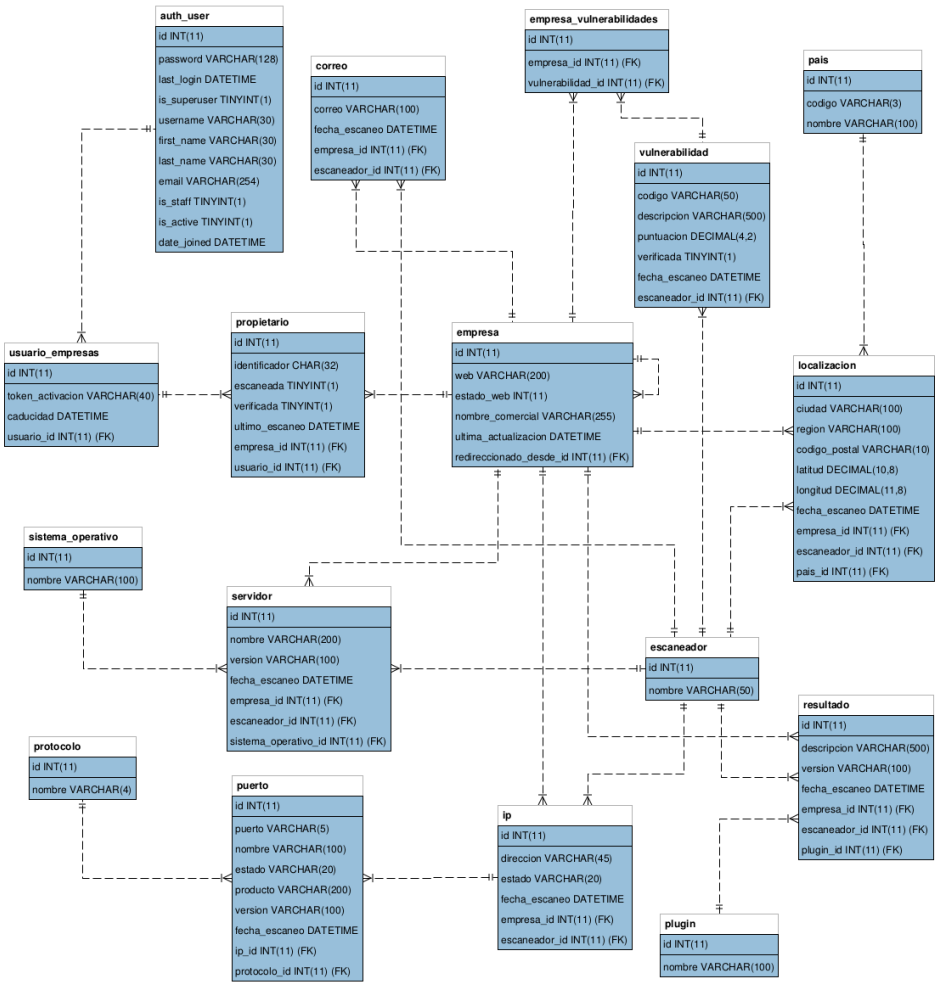


Figura 4.1: Diagrama de Entidad-Relación

Como se puede apreciar, la base de datos consta de 17 tablas, las cuales son:

- Escaneador
- Vulnerabilidad
- Empresa
- Usuario (tabla auth_user, autogenerada por Django)

- Usuario - Empresas
- Empresa - Vulnerabilidades
- Propietario
- Plugin
- Protocolo
- Sistema operativo
- Servidor
- IP
- País
- Localización
- Puerto
- Correo
- Resultado

4.2. Diagrama de clases

Al tratarse de una aplicación generada en base al framework Django (véase: Arquitectura), no se han definido clases más allá de las propias del modelo, aspecto que se detallará más adelante en la sección dedicada a la implementación; por ello, el proyecto funciona mediante una serie de scripts implementados en Python.

Únicamente se han añadido a las clases mencionadas dos nuevas, las cuales son *EmpresaForm* y *UsuarioForm* y que extienden de la clase que Django define para el uso de formularios, permitiendo la generación de elementos *form* de HTML con sus respectivas validaciones automatizadas, en base a los atributos seleccionados de una clase concreta del modelo.

En la Figura 4.2 se muestra el diagrama resultante.

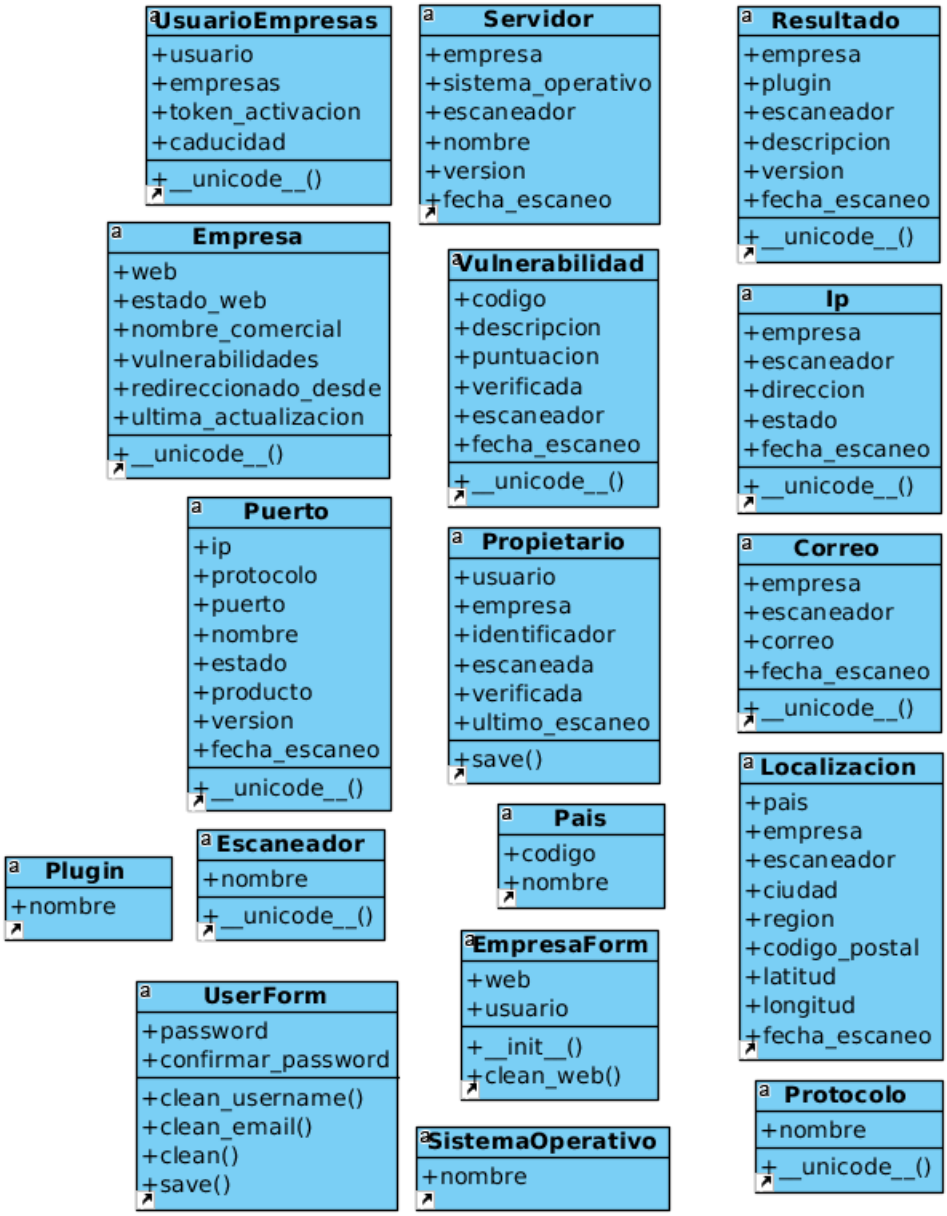


Figura 4.2: Diagrama de clases

4.3. Diagrama de actividades

Finalmente, en base a lo anteriormente expuesto, se ha decidido sustituir el concepto de diagrama de secuencia (definido como tarea en el EDT) por diagrama de actividades. Esta decisión viene dada debido a la naturaleza de Django y a que finalmente no se han implementado clases; en su lugar, se han definido una serie de scripts de diversa funcionalidad.

Se va a proceder a ilustrar en esta sección el diseño del método principal a implementar en el proyecto, debido a su importancia, al conformar la base de éste y estar reflejado en el EDT correspondiente al módulo de Captura, almacenamiento y análisis de resultados (véase: Alcance).

4.3.1. Escaneo

En la Figura 4.3 se expone el diagrama de actividades resultante y el cual hay que seguir para la implementación del método de escaneo.

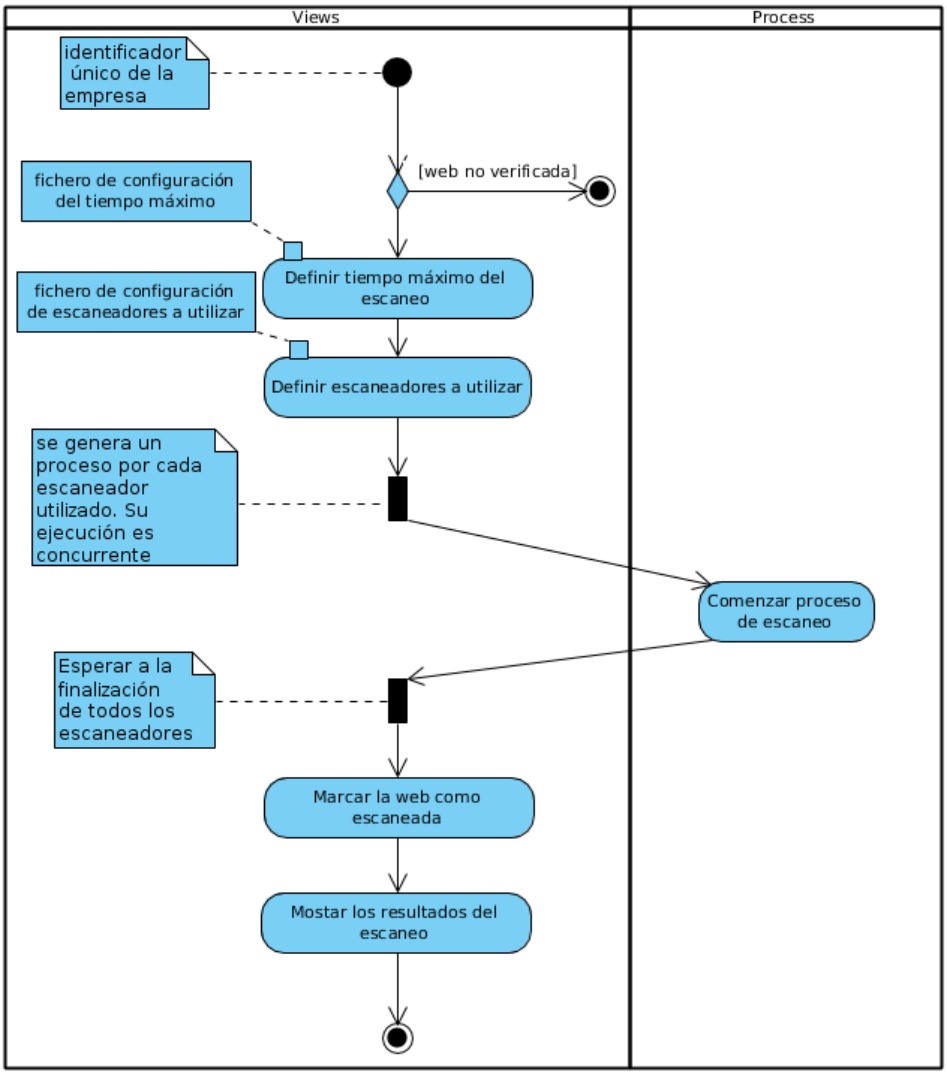


Figura 4.3: Diagrama de actividades del proceso de escaneo

Como puede apreciarse, el proceso de escaneo engloba la ejecución del conjunto de escaneadores implementados, que se lanzan de forma paralela y concurrente, haciendo uso de procesos de Python. Cada uno de éstos,

ejecuta una función específica, que escanea la web recibida y almacena sus resultados en la base de datos de la aplicación.

Además, nuevos escaneadores pueden ser incluídos en el proceso, ya que son cargados de forma dinámcia, mediante un fichero de configuración editable.

4.3.2. Script de captura y almacenamiento de resultados

Su funcionamiento es exactamente el mismo que se ha expuesto en la sección previa, con la única excepción de que, en lugar de cargar una única web mediante su identificador único, se descarga la totalidad de URLs contenidas en la base de datos y se van escaneando de forma secuencial, demorándose el proceso — en el peor de los casos — lo establecido en el fichero de configuración del tiempo máximo de escaneo individual multiplicado por el número de empresas almacenadas en el momento del escaneo global.

Se omite, por razones obvias, el último paso, el cual consiste en la muestra de resultados, al tratarse de un análisis masivo de sitios web.

4.3.3. Script de análisis de resultados

Para esta funcionalidad no se ve necesaria la realización de un diagrama en la que se plasme el flujo de tareas a seguir.

Funcionará de forma muy simple: se ejecutará periodicamente un script, en el que se realizará una llamada a una función por gráfico a generar, siendo cada uno para un propósito concreto (del análisis del lenguaje de programación utilizado en la generación de la web de cada empresa al tipo y versión del servidor de alojamiento). Estas funciones se ejecutarán de forma secuencial, obteniendo la información pertinente mediante llamadas a la base de datos y enviando ésta como entrada de la función externa encargada de la generación del gráfico.

5. Implementación

No se podía dar por concluido el presente documento sin comentar los aspectos más relevantes de la etapa fundamental del proyecto, la cual es su desarrollo. Este capítulo explicará el porqué de la elección de las herramientas utilizadas y reflejará las soluciones dadas a los contratiempos surgidos durante dicho periodo.

En primer lugar, se considera conveniente mencionar que, al tratarse de la continuación de un proyecto ya comenzado, el lenguaje de programación elegido, su correspondiente IDE, así como la utilización de frameworks, ya venían impuestos y no se consideró su reemplazo; uno de los principales motivos por los que quien escribe optó por la continuación del proyecto era precisamente por las tecnologías que se habían utilizado en su desarrollo.

5.1. Elección de APIs de los escaneadores

Esta ha sido la tarea que más se ha prolongado con respecto de la planificación inicial. La correcta elección de las herramientas de escaneo suponía la diferencia entre el éxito o fracaso del proyecto, tal y como se comenta en la documentación correspondiente a las anteriores versiones, en la que se exponía la ampliación de herramientas utilizadas como una posible mejora de cara a la continuación del desarrollo. A continuación se enumeran las APIs finalmente utilizadas, seleccionadas en base a dos premisas:

- que la información fuera devuelta en un **formato de salida JSON**, para su posterior tratamiento
- que el **rendimiento**, en tanto y cuanto a tiempo de ejecución, fuera asumible teniendo en cuenta el alto número de escaneos a realizar por el script de análisis a la base de datos

5.1.1. Uniscan

Antes de comenzar a explicar las herramientas utilizadas, conviene aclarar que, en relación a la primera herramienta heredada de la versión previa del proyecto, si bien en un comienzo decidió mantenerse, dando por hecho su utilidad, al cabo de un corto plazo de tiempo se decidió prescindir de ella, por las tres siguientes razones:

1. Un único escaneo se demoraba una media de cinco minutos.
2. Los resultados se almacenaban en un fichero de texto plano díficil de tratar.
3. No devolvía información relevante que otro escaneador no pudiera obtener en un tiempo menor.

5.1.2. Whatweb

Al igual que la anterior, el uso de esta herramienta proviene de la versión desde la que parte el presente proyecto. En este caso, se ha decidido mantener, si bien se ha modificado su utilización, en lo referente a la salida de los datos.

Originalmente, la información devuelta era almacenada en la base de datos en tablas que únicamente eran utilizadas por esta herramienta, duplicando información (p. ej., la URL de la web escaneada se almacenaba dos veces y no se establecía ninguna relación entre ambas, además de resultar redundante el simple hecho de realizar esta acción).

No obstante, la herramienta permitía definir que la salida de resultados se realizase en el formato estándar utilizado en el proyecto: JSON. Esto evita la necesidad de implementar un método que normalice la información ya almacenada en la base de datos, almacenándose directamente los resultados extraídos de la salida.

Una vez realizada esta modificación, se procedió a adaptar el código del método que escanea un sitio web mediante esta herramienta, almacenando de forma estandarizada la siguiente información:

- Código HTTP de respuesta del estado de la web.
- Dirección IP.
- Título del sitio web.

- Tipo y versión del servidor.
- País y ciudad en la que se encuentra alojado el servidor.
- Lista de correos electrónicos asociados a la web.

El resto de información se almacena como resultados adicionales.

5.1.3. nmap

La primera herramienta añadida en esta actualización venía a cubrir un vacío en el escaneo de los sitios web; hasta el momento no se estaba registrando información referente a los puertos. La API de esta herramienta devuelve un JSON con toda la información encontrada referente a puertos:

- Protocolo de transporte utilizado por el puerto.
- Estado, el cual indica si el puerto está abierto.
- Producto — así como versión del mismo — que se está ejecutando en dicho puerto.

5.1.4. Shodan

Probablemente la herramienta más potente utilizada en esta tercera versión, ya que la API informa de las vulnerabilidades halladas en la web escaneada, haciendo referencia al código CVE de cada una de ellas y devolviendo una descripción de la misma, así como si se puede verificar que dicha vulnerabilidad está afectando a la web o si únicamente es una sospecha, en base a la versión de alguna de las tecnologías utilizadas en dicha web.

5.1.5. What CMS

Sencillamente, se trata de una API que nos indica el CMS utilizado en la generación del sitio web escaneado y su versión, en caso de haberlo.

5.1.6. Who Hosts This

Al hilo de la anterior herramienta, esta hace lo propio con el *Internet Service Provider (ISP)* que aloja al servidor.

5.1.7. ipinfo

Esta herramienta devuelve información referente a geolocalización del servidor para el sitio web escaneado, tal que:

- País.
- Ciudad.
- Región.
- Código postal.
- Latitud.
- Longitud.

5.2. Base de datos

Habiendo trabajado siempre con bases de datos del tipo relacional, en general, y con MySQL en particular, se optó por seguir utilizándola para la presente fase del proyecto, por la preferencia de este tipo de bases de datos y sus características frente a las del tipo no-relacional u otras relacionales que, si bien más potentes, precisan de licencias de pago (como es el caso de Oracle).

El framework web Django posee además, una potente funcionalidad en su capa de persistencia de datos, anteriormente mencionada, que éste denomina como modelo. Lo que esto permite es realizar una técnica llamada mapeo objeto relacional (del inglés *Object-Relational mapping (ORM)*), la cual consiste en definir las tablas que van a conformar la base de datos como objetos del lenguaje de programación que se está utilizando en la implementación (en este caso, Python), haciendo uso de características propias de la orientación a objetos, como son la definición de atributos, métodos, herencias y polimorfismo. Posteriormente, mediante la ejecución de un comando en terminal, se extrae la información del modelo y se genera, de forma automática, la base de datos en base a los objetos definidos.

El Object-Relational mapping implementa adicionalmente potentes métodos de encapsulamiento necesarios para la ejecución de sentencias DML, liberando al programador de la implementación de dichos métodos. Es por ello, que se ha definido toda la lógica en el propio modelo, en lugar de definir procedimientos almacenados o *triggers* en la propia base de datos, lo

cual facilita y deja el entorno preparado para la posible tarea de migración a otro tipo de base de datos, si el proyecto así lo requiere.

Cabe mencionar, que en cierto punto del desarrollo se planteó la idea de migración de la base de datos a *MongoDB*, sistema de base de datos orientada a documentos, al igual que JSON, formato estándar utilizado en la captura de resultados de los diferentes escaneadores. Esta idea fue finalmente desechada, debido a la complejidad en el aprendizaje de una nueva tecnología una vez empezada la fase de implementación del proyecto, unido a que resultaba necesaria la realización de un nuevo diagrama de base de datos, ya que un Diagrama de Entidad-Relación no se ajustaba al concepto de base de datos no-relacional.

5.3. Ficheros de configuración

Se han definido para el proyecto dos ficheros de texto plano que contienen información editable de la que se nutre el algoritmo de escaneo. La finalidad de estos es permitir la modificación de una serie de parámetros, descritos a continuación, sin necesidad de modificar el código fuente de la aplicación y, por lo tanto, sin obligar a un reinicio y nuevo despliegue de la aplicación a fin de recoger las modificaciones realizadas.

5.3.1. Escaneadores seleccionados

Permite la inserción de los nombres de los escaneadores a ejecutar. La aplicación procede a la lectura del fichero y, tras comprobar que el escaneador está registrado en la base de datos e implementado su método correspondiente, utiliza dicha función en el escaneo actual.

Si en un futuro se decide eliminar la ejecución de algún escaneador en el proceso, este fichero permitirá definir dicho comportamiento. Del mismo modo, si se implementa un nuevo escaneador, habrá que identificarlo en este fichero (además de haber definido la función pertinente y añadido éste a la base de datos).

5.3.2. Tiempo máximo de ejecución por escaneo

Establece un tiempo límite, en segundos, que indica cuánto se puede demorar el proceso de escaneo, permitiendo mediante este valor eliminar del proceso los escaneadores lentos o solventar problemas de ejecución en caso de fallo de uno de los componentes, sin afectar al resto de ellos.

5.4. Interfaz de usuario web

Finalmente, satisfaciendo los RIE definidos previamente (véase: Requisitos específicos) y permitiendo un ahorro de tiempo considerable al no precisar una implementación desde cero, se ha optado por la utilización del framework web frontend *Bootstrap*, que implementa las hojas de estilo en cascada (CSS) de los componentes básicos de una página web, con un diseño responsivo.

Se utilizaron diversos prototipos a papel para el diseño inicial del sitio web, el cual se fue modificando, en base a propuestas de personas ajenas al proyecto a las que se pedía opinión tras el uso de la herramienta.

5.5. Generación de gráficos interactivos

Para dicha tarea, tal y como se ha mencionado previamente, se ha hecho uso de la herramienta **plot.ly**, la cual es compatible con Python. Los gráficos generados en base a la información contenida en la base de datos son almacenados como código HTML, dentro de una etiqueta *div*, a la cual se realiza una llamada desde la página de *Datos de interés*, tras lo cual se incrusta en dicha página de la aplicación el gráfico pertinente, para su visualización por parte del usuario final.

5.6. Automatización de procesos

Para la automatización de tareas a realizar por el servidor, se ha optado por el uso de la herramienta **crontab**, disponible para su configuración en el propio sistema operativo Linux de éste. En ella, se indica el patrón de ejecución a seguir, pudiendo especificar los segundos, minutos, horas, días del mes, meses, días de la semana y años en los que ha de lanzarse la tarea programada.

Siguiendo este procedimiento, se han definido tres procesos, explicados a continuación:

- **Copia de seguridad de la base de datos**. Diariamente, se realizará el volcado de la información contenida en la base de datos a un fichero, a fin de poder recuperar la información en caso de fallo en el sistema.
- **Captura y almacenamiento de información**. Mensualmente se ejecutará el script de escaneo global de sitios web contenidos en la base

de datos, almacenando la fecha de cada escaneo, a fin de mantener un histórico de resultados.

- **Análisis de resultados**. De forma análoga, se ejecutará semanalmente el script de generación de gráficos en base a la información contenida en ese momento en la base de datos.

5.7. Estructura de ficheros

Para concluir, la estructura de ficheros del proyecto ha quedado organizada tal que:

- euskalcrawlerProject
  - euskalcrawler
    - admin.py
    - backends.py
    - forms.py
    - models.py
    - tests.py
    - urls.py
    - views.py
  - euskalcrawlerProject
    - settings.py
    - urls.py
    - wsgi.py
  - results
  - scans
    - max_tiempo.txt
    - scans.txt
    - webs.txt
  - static
    - css
    - img
    - js
  - templates
    - euskalcrawler

        - graficos
- actualiza_graficos.py
- escaneo_empresas.py
- manage.py
- populate_euskalcrawler.py

6. Resultados

En este capítulo, previo a la conclusiones finales del proyecto, se ilustran en sus diferentes formas las validaciones realizadas para ratificar el cumplimiento de la ERS y, por ende, de los objetivos iniciales.

Se expondrán, en primer lugar, las pruebas realizadas a fin de verificar el correcto funcionamiento de la aplicación desarrollada para, posteriormente, reflejar en forma de gráficos los resultados alcanzados tras la captura y análisis de información, obtenida del proceso de escaneo global de la base de datos del proyecto.

6.1. Pruebas de verificación de los requisitos

Cabe comentar en primer lugar, que las pruebas se han realizado parcialmente — y en la medida de lo posible — de forma automatizada, por medio del fichero *tests* que Django proporciona para este fin. Siguiendo esta metodología se han cubierto las principales funcionalidades de la aplicación, enteniendose por principales aquellas con las que el usuario final interactúa de forma directa.

Adicionalmente y a lo largo de la fase de implementación proyecto, se han realizado de forma manual pruebas en tiempo real de las funcionalidades desarrolladas, asegurando en todo momento la correcta ejecución de los métodos implementados en el sistema. Si bien este procedimiento cubre la casuística total de la foto actual de la aplicación, un cambio en la implementación conllevaría a incertidumbre en el resultado para las mismas pruebas.

A continuación, se describe cómo se ha llevado a cabo el proceso mencionado, desglosado por requisitos específicos del ERS.

6.1.1. Verificación de los requisitos funcionales

RF1. Alta de usuario

Comprobación del correcto etiquetado de los campos en el formulario de registro
Descripción: Se asevera que el texto de las etiquetas describe su contexto. **Resultado esperado**: El texto de etiquetado mostrado es el esperado.
Validación: Éxito.

Tabla 6.1: Comprobación del correcto etiquetado de los campos en el formulario de registro

Comprobación de la obligatoria inserción del nombre de usuario en el formulario de registro
Descripción: Se asevera que el campo nombre de usuario del formulario es requisito para el correcto procesamiento de la operación de registro de nuevo usuario en la aplicación, realizando un intento de alta sin introducir este campo. **Resultado esperado**: El formulario no pasa la validación previa al envío de datos.
Validación: Éxito.

Tabla 6.2: Comprobación de la obligatoria inserción del nombre de usuario en el formulario de registro

Comprobación de la obligatoria inserción del email en el formulario de registro
Descripción: Se asevera que el campo email del formulario es requisito para el correcto procesamiento de la operación de registro de nuevo usuario en la aplicación, realizando un intento de alta sin introducir este campo. **Resultado esperado**: El formulario no pasa la validación previa al envío de datos.
Validación: Éxito.

Tabla 6.3: Comprobación de la obligatoria inserción del email en el formulario de registro

Comprobación de la obligatoria inserción de la contraseña en el formulario de registro
Descripción: Se asevera que los campos constraseña y confirmar contraseña del formulario son requisito para el correcto procesamiento de la operación de registro de nuevo usuario en la aplicación, realizando un intento de alta sin introducir dichos campos.
Resultado esperado: El formulario no pasa la validación previa al envío de datos.
Validación: Éxito.

Tabla 6.4: Comprobación de la obligatoria inserción de la contraseña en el formulario de registro

Comprobación de la inserción de un nombre de usuario no existente en el sistema en el formulario de registro
Descripción: Se asevera que el campo nombre de usuario del formulario ha de ser único en el sistema para el correcto procesamiento de la operación de registro de nuevo usuario en la aplicación, realizando un intento de alta introduciendo un valor ya existente para este campo.
Resultado esperado: El formulario no pasa la validación previa al envío de datos.
Validación: Éxito.

Tabla 6.5: Comprobación de la inserción de un nombre de usuario no existente en el sistema en el formulario de registro

Comprobación de la inserción de un email no existente en el sistema en el formulario de registro
Descripción: Se asevera que el campo email del formulario ha de ser único en el sistema para el correcto procesamiento de la operación de registro de nuevo usuario en la aplicación, realizando un intento de alta introduciendo un valor ya existente para este campo.
Resultado esperado: El formulario no pasa la validación previa al envío de datos.
Validación: Éxito.

Tabla 6.6: Comprobación de la inserción de un email no existente en el sistema en el formulario de registro

Comprobación de la obligatoria inserción de la confirmación de contraseña en el formulario de registro
Descripción: Se asevera que el campo contraseña del formulario ha de estar en concordancia con el campo confirmar contraseña para el correcto procesamiento de la operación de registro de nuevo usuario en la aplicación, realizando un intento de alta en el que estos campos tomen valores distintos. **Resultado esperado**: El formulario no pasa la validación previa al envío de datos.
Validación: Éxito.

Tabla 6.7: Comprobación de la inserción de una contraseña confirmada de forma errónea en el formulario de registro

Comprobación del alta de un nuevo usuario sin errores mediante el formulario de registro
Descripción: Se asevera que la operación de registro de un usuario nuevo en la aplicación se realiza de forma correcta para unos datos no erróneos. **Resultado esperado**: El formulario pasa la validación previa al envío de datos.
Validación: Éxito.

Tabla 6.8: Comprobación del alta de un nuevo usuario sin errores mediante el formulario de registro

Comprobación del envío de correo de confirmación del alta de un nuevo usuario
Descripción: Se comprueba que tras una correcta operación de registro de un usuario nuevo en la aplicación se realiza el envío de un correo de confirmación a la dirección introducida en el formulario.
textbfResultado esperado: Mensaje con enlace de confirmación recibido en la cuenta de correo del nuevo usuario.
Validación: Éxito.

Tabla 6.9: Comprobación del envío de correo de confirmación del alta de un nuevo usuario

RF2. Acceso al Sistema

Comprobación de la autenticación mediante inserción del nombre de usuario en el formulario de identificación
Descripción: Se comprueba la correcta identificación del usuario mediante la inserción de un nombre de usuario registrado en el sistema en el formulario. **Resultado esperado**: El usuario accede al sistema por medio de su nombre de usuario y contraseña.
Validación: Éxito.

Tabla 6.10: Comprobación de la autenticación mediante inserción del nombre de usuario en el formulario de identificación

Comprobación de la autenticación mediante inserción del email en el formulario de identificación
Descripción: Se comprueba la correcta identificación del usuario mediante la inserción de una dirección de correo electrónico registrada en el sistema en el formulario. **Resultado esperado**: El usuario accede al sistema por medio de su email y contraseña.
Validación: Éxito.

Tabla 6.11: Comprobación de la autenticación mediante inserción del email en el formulario de identificación

Comprobación del error en autenticación mediante inserción de datos no registrados en el formulario de identificación
Descripción: Se comprueba que, para cualquiera de los campos del formulario, si se introduce un dato de identificación no registrado en el sistema en el formulario no se autoriza el acceso a la aplicación. **Resultado esperado**: El usuario no es autorizado a acceder al sistema.
Validación: Éxito.

Tabla 6.12: Comprobación del error en autenticación mediante inserción de datos no registrados en el formulario de identificación

RF3. Desconexión de usuario

Comprobación de la correcta desconexión de un usuario autenticado en el sistema
Descripción: Se comprueba la correcta implementación de la desconexión de un usuario autenticado e identificado en el sistema. **Resultado esperado**: La sesión del usuario finaliza y se revoca su autorización a navegar por su área personal hasta su nueva autenticación.
Validación: Éxito.

Tabla 6.13: Comprobación de la correcta desconexión de un usuario autenticado en el sistema

RF4. Consulta de sitios web

Comprobación de la visualización del área personal del usuario sin empresas registradas
Descripción: Se comprueba que la página de área personal del usuario no muestra ninguna empresa registrada. **Resultado esperado**: Página de área personal del usuario vacía.
Validación: Éxito.

Tabla 6.14: Comprobación de la visualización del área personal del usuario sin empresas registradas

Comprobación de la visualización del área personal del usuario con empresa registrada y no verificada
Descripción: Se comprueba que la página de área personal del usuario muestra la empresa registrada con los pasos a seguir para su verificación. **Resultado esperado**: Página de área personal del usuario con la empresa registrada y el contenido (identificador único) a generar para la verificación de ésta.
Validación: Éxito.

Tabla 6.15: Comprobación de la visualización del área personal del usuario con empresa registrada y no verificada

Comprobación de la visualización del área personal del usuario con empresa verificada y no escaneada
Descripción: Se comprueba que la página de área personal del usuario muestra la empresa registrada con el botón a pulsar para el escaneo de ésta. **Resultado esperado**: Página de área personal del usuario con la empresa registrada y el botón de escaneo habilitado para su ejecución.
Validación: Éxito.

Tabla 6.16: Comprobación de la visualización del área personal del usuario con empresa verificada y no escaneada

Comprobación de la visualización del área personal del usuario con empresa escaneada sin vulnerabilidades detectadas
Descripción: Se comprueba que la página de área personal del usuario muestra la empresa registrada con el botón a pulsar para mostrar los resultados del escaneo de ésta, indicando que no se han detectado vulnerabilidades. **Resultado esperado**: Página de área personal del usuario con la empresa registrada, el mensaje indicativo de la ausencia de vulnerabilidades y el botón de resultados habilitado para su ejecución.
Validación: Éxito.

Tabla 6.17: Comprobación de la visualización del área personal del usuario con empresa escaneada sin vulnerabilidades detectadas

Comprobación de la visualización del área personal del usuario con empresa escaneada con vulnerabilidades detectadas
Descripción: Se comprueba que la página de área personal del usuario muestra la empresa registrada con el botón a pulsar para mostrar los resultados del escaneo de ésta, indicando el número de vulnerabilidades que se han detectado. **Resultado esperado**: Página de área personal del usuario con la empresa registrada, el mensaje indicativo del número de vulnerabilidades detectadas y el botón de resultados habilitado para su ejecución.
Validación: Éxito.

Tabla 6.18: Comprobación de la visualización del área personal del usuario con empresa escaneada con vulnerabilidades detectadas

Comprobación de la visualización del área personal del usuario con más de una empresa registrada
Descripción: Se comprueba que la página de área personal del usuario muestra la lista de empresa registradas con los datos correspondientes al estado para cada una de ellas. **Resultado esperado**: Página de área personal del usuario con la lista de empresas registradas e información correspondiente a cada una de ellas.
Validación: Éxito.

Tabla 6.19: Comprobación de la visualización del área personal del usuario con más de una empresa registrada

RF5. Alta de sitio web

Comprobación del correcto etiquetado de los campos en el formulario de registro de empresa
Descripción: Se asevera que el texto de las etiquetas describe su contexto. **Resultado esperado**: El texto de etiquetado mostrado es el esperado.
Validación: Éxito.

Tabla 6.20: Comprobación del correcto etiquetado de los campos en el formulario de registro de empresa

Comprobación de la obligatoria inserción de la web en el formulario de registro de empresa
Descripción: Se asevera que el campo web del formulario es requisito para el correcto procesamiento de la operación de registro de la web de una empresa en la aplicación, realizando un intento de alta sin introducir este campo. **Resultado esperado**: El formulario no pasa la validación previa al envío de datos.
Validación: Éxito.

Tabla 6.21: Comprobación de la obligatoria inserción de la web en el formulario de registro de empresa

Comprobación de la inserción de una web no registrada para el usuario conectado en el formulario de registro de empresa (Test 1)
Descripción: Se asevera que el campo web del formulario ha de ser único para el usuario conectado para el correcto procesamiento de la operación de registro de la web de una empresa en la aplicación, realizando un intento de alta introduciendo un valor ya existente para este campo. **Resultado esperado**: El formulario no pasa la validación previa al envío de datos.
Validación: Éxito.

Tabla 6.22: Comprobación de la inserción de una web no registrada para el usuario conectado en el formulario de registro (Test 1)

Comprobación de la inserción de una web no registrada para el usuario conectado en el formulario de registro de empresa (Test 2)
Descripción: Se asevera que el campo web del formulario ha de ser único para el usuario conectado con independencia de otros usuarios — que en este supuesto pueden disponer de la misma web registrada — para el correcto procesamiento de la operación de registro de la web de una empresa en la aplicación, realizando un intento de alta introduciendo un valor ya existente para este campo. **Resultado esperado**: El formulario no pasa la validación previa al envío de datos.
Validación: Éxito.

Tabla 6.23: Comprobación de la inserción de una web no registrada para el usuario conectado en el formulario de registro (Test 2)

Comprobación del alta de una nueva web de empresa sin errores mediante el formulario de registro de empresa
Descripción: Se asevera que la operación de registro de una nueva web de empresa en la aplicación se realiza de forma correcta para unos datos no erróneos. **Resultado esperado**: El formulario pasa la validación previa al envío de datos.
Validación: Éxito.

Tabla 6.24: Comprobación del alta de una nueva web de empresa sin errores mediante el formulario de registro de empresa

RF6. Verificación de sitio web

Comprobación de la verificación automática satisfactoria de la autoría de una empresa
Descripción: El sistema comprueba que una página web corporativa ha sido verificada por el usuario identificado en el sistema. **Resultado esperado**: Se verifica la autoría del sitio web y se permite su escaneo.
Validación: Éxito.

Tabla 6.25: Comprobación de la verificación automática de la autoría de una empresa

Comprobación de la verificación automática insatisfactoria de la autoría de una empresa
Descripción: El sistema comprueba que una página web corporativa no ha sido verificada por el usuario identificado en el sistema. **Resultado esperado**: No se habilita el escaneo del sitio web no verificado, proporcionando la información necesaria para la realización de dicha operación al usuario identificado.
Validación: Éxito.

Tabla 6.26: Comprobación de la verificación automática insatisfactoria de la autoría de una empresa

RF7. Escaneo de sitio web

Comprobación del escaneo de un sitio web desde el área personal del usuario con empresa verificada
Descripción: Se comprueba que la página de área personal del usuario permite y realiza la operación de escaneo de sitio web para una empresa verificada. **Resultado esperado**: La web verificada es escaneada, almacenando sus resultados.
Validación: Éxito.

Tabla 6.27: Comprobación del escaneo de un sitio web desde el área personal del usuario con empresa verificada

Comprobación del escaneo de un sitio web desde el área personal del usuario con empresa ya escaneada
Descripción: Se comprueba que la página de área personal del usuario permite y realiza la operación de escaneo de sitio web para una empresa que ya ha sido escaneada previamente. **Resultado esperado**: La web verificada es escaneada, almacenando sus resultados y manteniendo un histórico en la base de datos.
Validación: Éxito.

Tabla 6.28: Comprobación del escaneo de un sitio web desde el área personal del usuario con empresa ya escaneada

Un objetivo a futuro, que no ha podido implementarse a día de hoy por falta de tiempo, sería la inclusión de métodos que realicen comprobaciones individuales para cada herramienta escaneadora, contenidos en el mencionado fichero *tests*. No obstante, todos los métodos han sido verificados para diferentes casos de prueba, de los que se conocía el resultado de antemano, además de tratarse de APIs desarrolladas por terceros, cuyo correcto funcionamiento ha sido previamente validado por los desarrolladores encargados de su implementación y que constituye la condición *sine qua non* para su uso en el presente proyecto.

RF8. Consulta de resultados de sitio web

Comprobación de la visualización de los resultados de un escaneo a una empresa registrada
Descripción: Se comprueba el acceso a la página de resultados para una empresa registrada y escaneada. **Resultado esperado**: Página de resultados del usuario con la lista de empresas registradas e información correspondiente a cada una de ellas.
Validación: Éxito.

Tabla 6.29: Comprobación de la visualización del área personal del usuario con más de una empresa registrada

RF9. Consulta de datos de intereś

Comprobación de la visualización de los datos de interés para un usuario no autenticado en el sistema
Descripción: Se comprueba el acceso a la página de datos de interés para un usuario no autenticado en el sistema, mostrados en la página principal. **Resultado esperado**: La página principal muestra una lista de datos de interés que pueden ser consultados sin previo registro.
Validación: Éxito.

Tabla 6.30: Comprobación de la visualización de los datos de interés para un usuario no autenticado en el sistema

Comprobación de la visualización de los datos de interés para un usuario autenticado en el sistema
Descripción: Se comprueba el acceso a la página de datos de interés para un usuario autenticado en el sistema, mostrados en la página principal. **Resultado esperado**: Página de datos de interés con lista de gráficos a consultar disponible para un usuario autenticado en el sistema.
Validación: Éxito.

Tabla 6.31: Comprobación de la visualización de los datos de interés para un usuario autenticado en el sistema

RF10. Escaneo automático

Para los scripts encargados de la captura y almacenamiento de resultados, así como de análisis de éstos y generación de gráficos, se generaron dos ficheros de log: uno para registrar la traza de ejecución, a fin de analizar a posteriori si el funcionamiento de éstos ha sido el esperado, así como otro para captura de errores, con el objetivo de que, a la finalización de la ejecución este fichero no tuviese contenido, lo cual indica que no ha habido ningún error de ejecución. Estos ficheros analizaron, en su última ejecución, un total de 3088 registros referentes a empresas sin errores.

6.1.2. Verificación de los requisitos de interfaces externos

RIE1. Compatibilidad con navegadores

Se ha comprobado el correcto funcionamiento, en cuanto a la totalidad de RFs se refiere, en todos los navegadores definidos inicialmente. La respuesta ha sido satisfactoria para todos los casos, utilizando un sistema operativo Ubuntu 16.04 para Google Chrome y Mozilla Firefox y un sistema operativo Windows 7 para Internet Explorer.

A continuación, se muestran las diferentes páginas que componen la aplicación web.

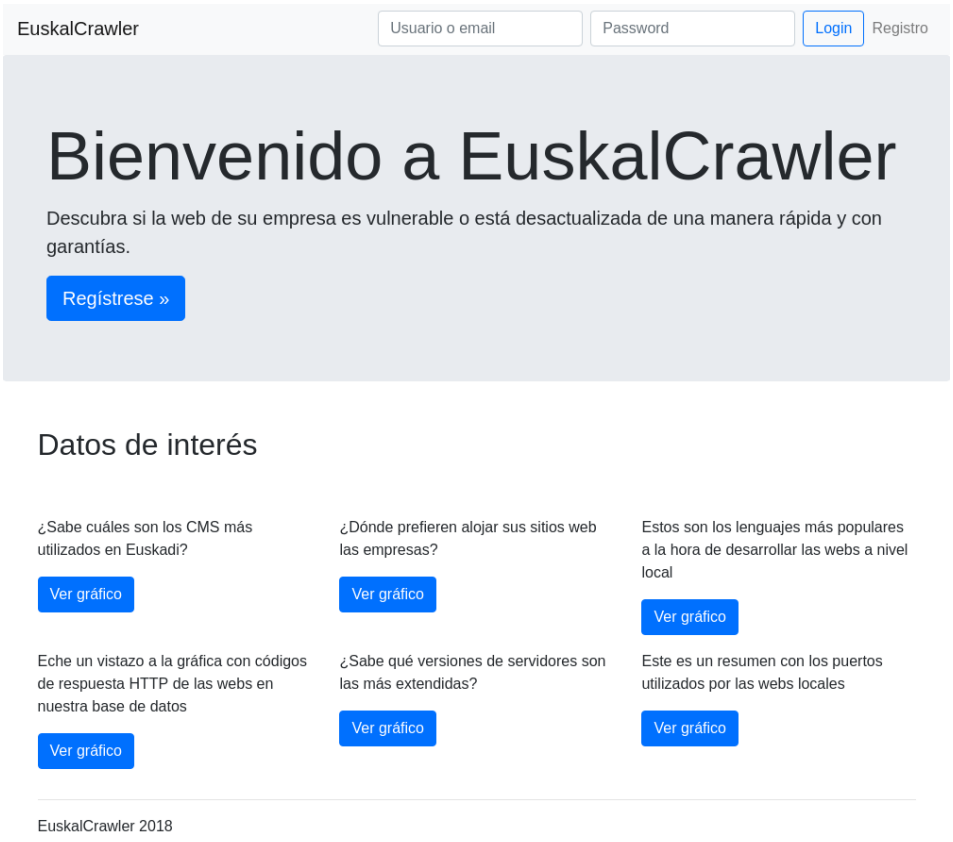


Figura 6.1: Página principal

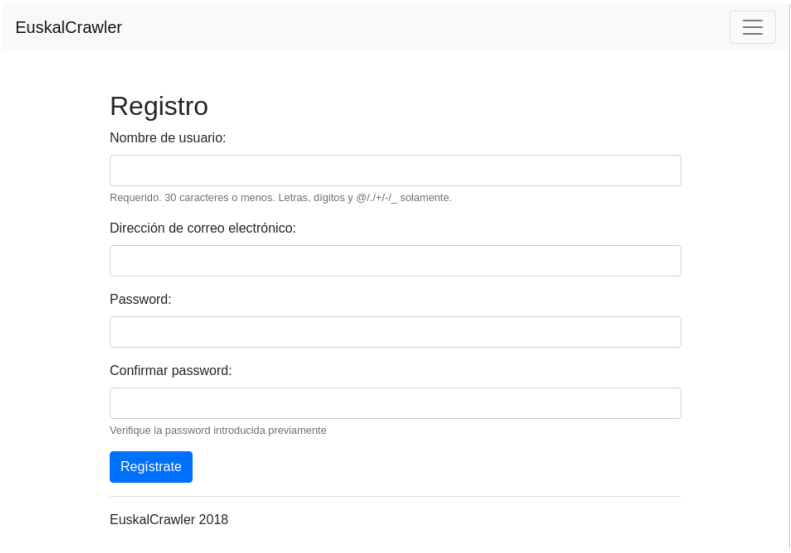


Figura 6.2: Página de registro

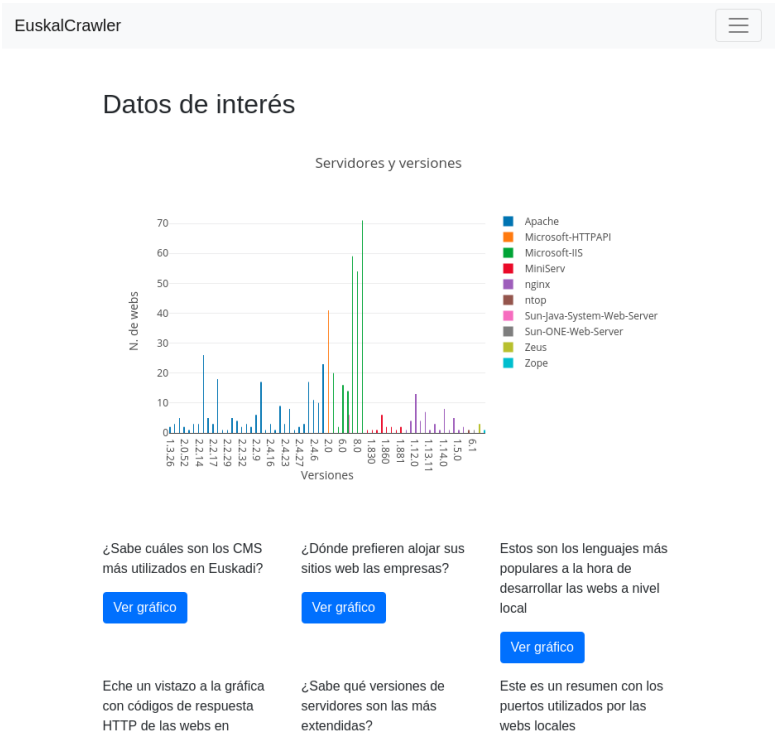


Figura 6.3: Página de datos de interés (gráficos)

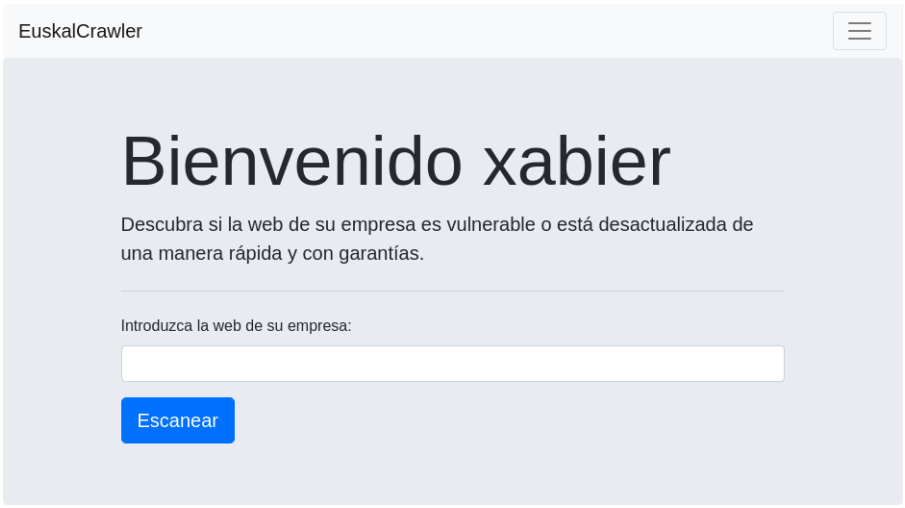


Figura 6.4: Página personal del usuario (parte superior)

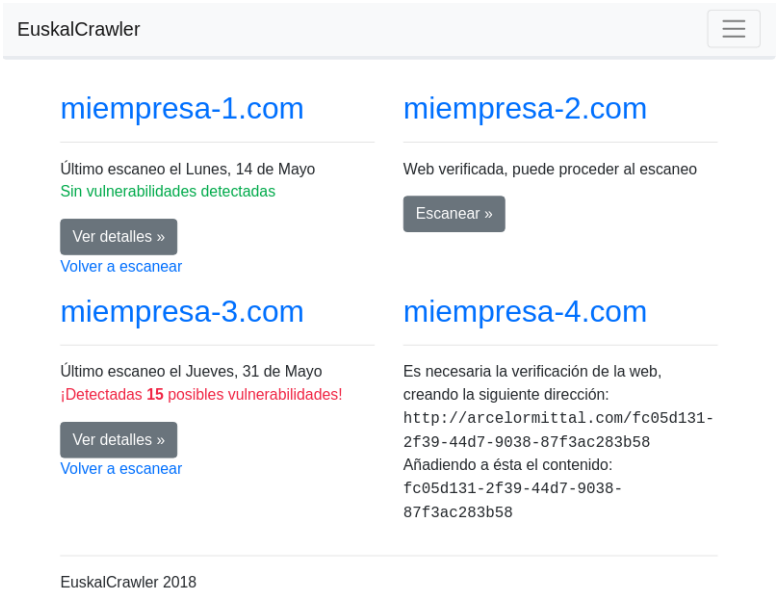


Figura 6.5: Página personal del usuario (parte inferior)

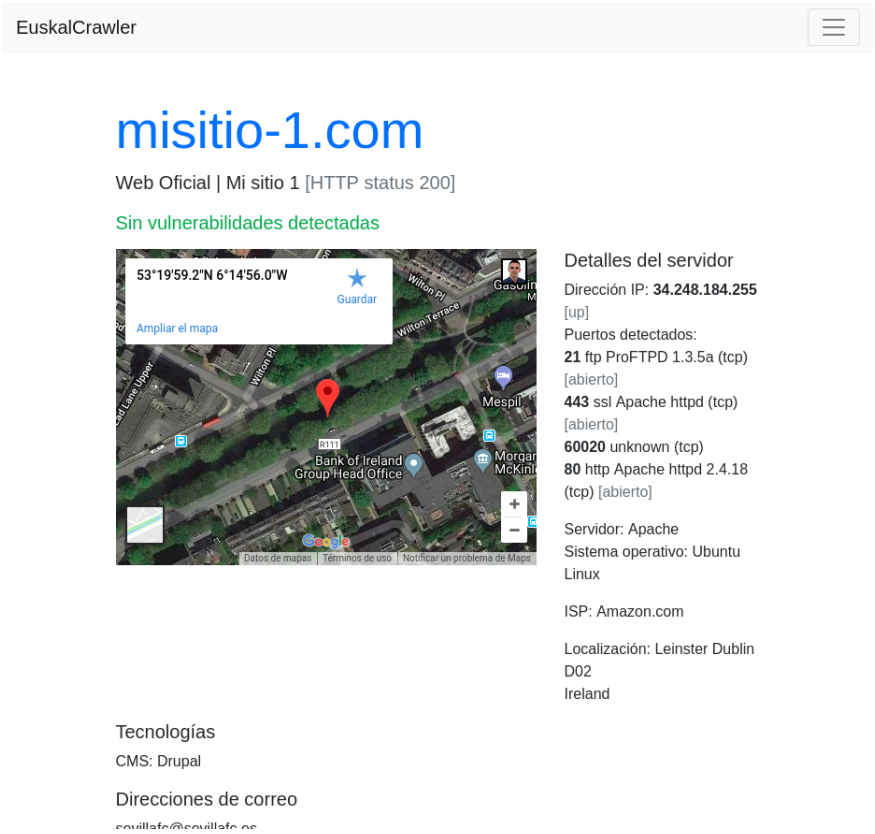


Figura 6.6: Página de resultados

RIE2. Diseño responsivo

La inclusión del framework frontend *Bootstrap* para la implementación de las hojas de estilo en cascada ha supuesto la cumplimentación de este requisito[1].

Adicionalmente, para asegurar la compatibilidad, se han realizado pruebas del sistema, tanto con un smartphone con sistema operativo Android, como mediante la consola de desarrollo del navegador Google Chrome, que permite emular diferentes dispositivos móviles.

[1] Navegadores y dispositivos compatibles con *Bootstrap 4.1*: https://getbootstrap.com/docs/4.1/getting-started/browsers-devices/

6.2. Análisis de los resultados finales

Tras exponer las diferentes pruebas realizadas, evidenciando por medio de ellas el correcto funcionamiento del sistema, se puede afirmar que éste se encuentra preparado para su uso, así como que los resultados de las operaciones para las cuales se ha programado son veraces, satisfaciendo los requisitos definidos.

Partiendo de esta premisa, a continuación se procede a comentar los resultados del primer escaneo completo a la base de datos de empresas almacenadas hasta el momento — con un total de tres mil ochenta y ocho sitios web registrados —, que nos dan una idea tan realista como concreta de las principales tendencias en el sector de desarrollo web local.

Acompañarán a estos resultados los gráficos originales, autogenerados mediante el script periódico de análisis, que se encuentran disponibles y pueden ser consultados desde la propia web del proyecto.

6.2.1. Sistemas de gestión de contenidos (CMS) y versiones

El primer gráfico, extrae de la base de datos y muestra la información para la que inicialmente fue concebido el proyecto; tal y como se aprecia en la Figura 6.7, éste refleja los diferentes sistemas de gestión de contenidos utilizados para la creación del computo total de páginas web registradas en el sistema, en los casos en que se haya partido de un CMS para facilitar el desarrollo. Por tanto, y como es lógico, las cifras totales de este gráfico no alcanzan el número previamente comentado, no encontrándose una versión de CMS utilizada para una cifra superior a doscientos doce sitios web.

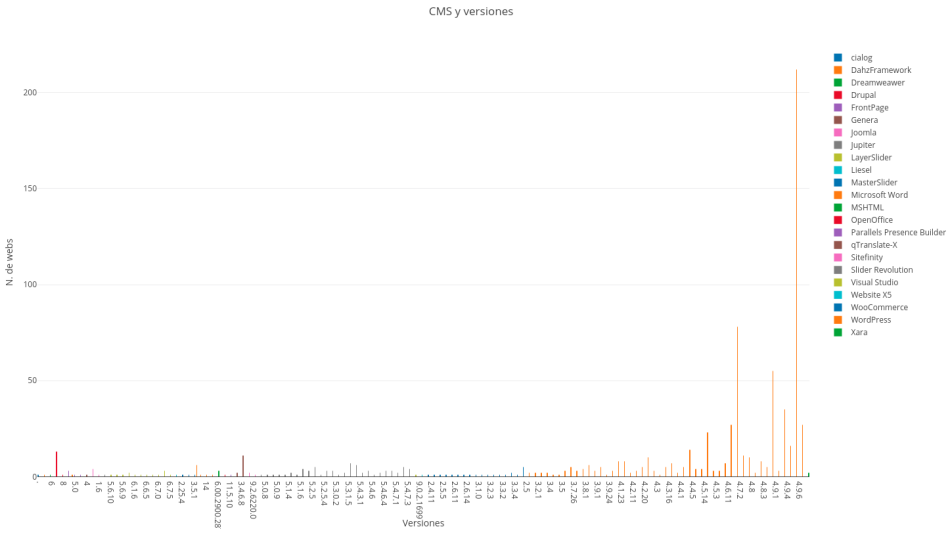


Figura 6.7: Gráfico de CMS y versiones

En primer lugar, cabe comentar que el eje ordenadas representa el número total de páginas que utilizan cada una de las tecnologías, representadas a su vez en el eje de abscisas, en forma de versiones. El total de versiones de cada CMS es mostrado en un mismo color, para facilitar una rápida identificación visual.

Comentando los resultados del gráfico, salta a la vista la clara y abismal ventaja de *WordPress* sobre el resto de herramientas, abarcando alrededor de un tercio del gráfico, en lo que a versiones se refiere. No es de asombrar, si se repara en que esta es una solución de software utilizada mundialmente por aproximadamente veintisiete millones de sitios web, con una cuota de

mercado del 93 % [2].

Mostrando únicamente los resultados que implican la utilización de *WordPress*, se advierte que las empresas que utilizan esta tecnología parecen ser conscientes de los problemas que puede implicar el uso de una versión no actualizada de un producto, siendo la versión 4.9.5 del producto la más utilizada, con una diferencia del orden de 134 unidades con respecto a la siguiente versión con uso más expandido (la última versión estable es la 4.9.7, publicada hace menos de una quincena de días, motivo por el cual aún no se muestra en el gráfico ninguna web que haga uso de ésta). La siguiente versión más utilizada es la 4.7.10, la cual data del 3 de abril de 2018, por lo que, en términos generales, se puede afirmar que las páginas se encuentran al día.

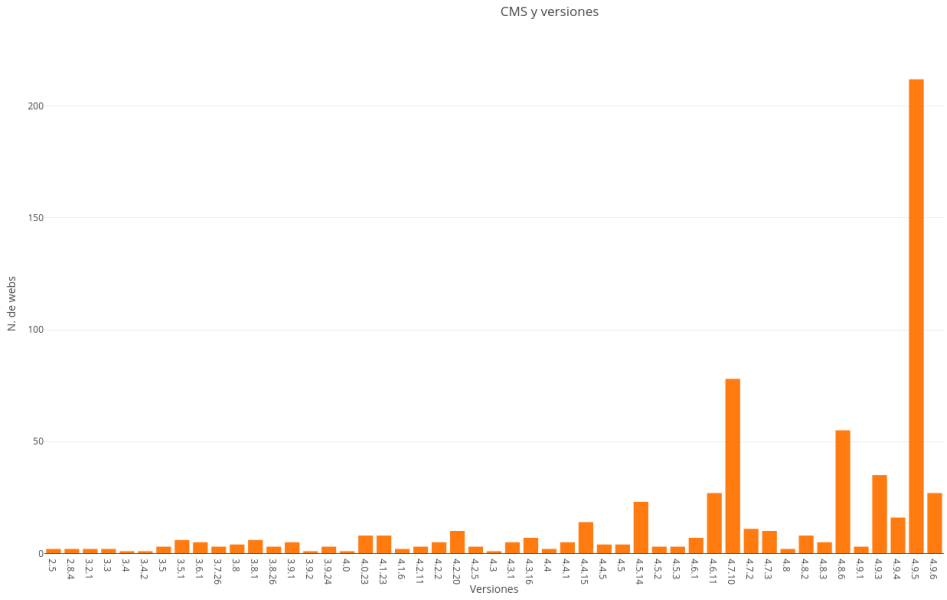


Figura 6.8: Gráfico de CMS y versiones. Wordpress

Por otra parte, para el resto de tecnologías no se ha detectado versión alguna que supere la docena de sitios web, tal como refleja la Figura 6.7. Encabezan la lista *Slider Revolution*, *Drupal*, *Joomla*, *qTranslate-X*, *LayerSlider* y *WooCommerce*, si bien la mayoría no son más que complementos basados nuevamente en *WordPress*, que añaden funcionalidades muy es-

[2]Datos extraídos de: `https://trends.builtwith.com/cms`

pecíficas a éste, muy probablemente utilizadas para casos excepcionales en que el cliente impone requisitos muy concretos, que requieren desarrollos a medida.

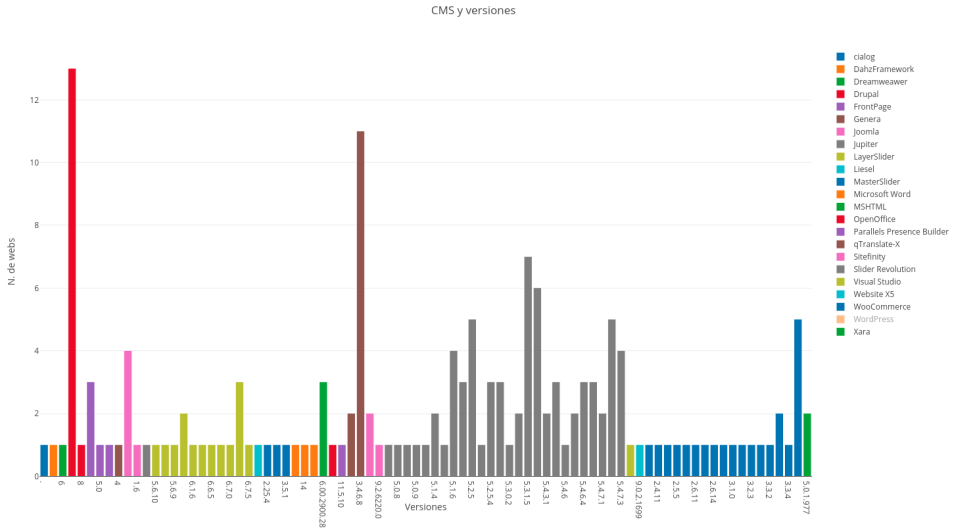


Figura 6.9: Gráfico de CMS y versiones. Resto de tecnologías

Por último, se precisa mostrar especial atención a las más que posibles vulnerabilidades en los sitios web que hacen uso de versiones obsoletas, así como en los casos aislados en que esta tarea ha recaído en herramientas no destinadas a realizar este trabajo, como es el caso de las empresas que han utilizado *Microsoft Word* para el desarrollo de su sitio web corporativo, en lo que refleja una decisión cuando menos cuestionable.

6.2.2. Internet Service Provider (ISP)

Este segundo gráfico estudia las tendencias de las empresas en lo que respecta a la elección y contratación del proveedor de servicios de Internet, para los sitios web que disponen o facilitan la extracción de dicha información. El resultado final se expone mediante la Figura 6.10. Nuevamente, el eje horizontal representa los diferentes proveedores detectados, mientras que el eje vertical representa el número de empresas que cuentan con los servicios de dicho proveedor.

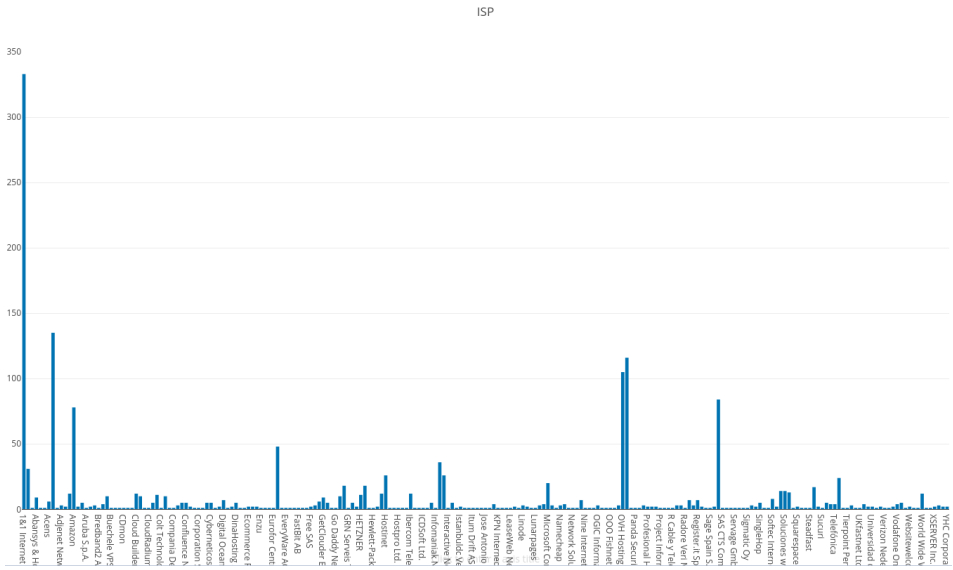


Figura 6.10: Gráfico de ISP

El liderazgo de *1&1* en este ámbito es prácticamente total, duplicando en cifras a sus competidores directos y seguido por *Acens*, *OVH* y *Sarenet*. El primer operador de telefonía conocido de la lista de resultados es, lógicamente, *Euskaltel*, al tratarse de un estudio a nivel local del cual es líder en cuota de mercado.

6.2.3. Lenguajes de programación

Un punto interesante del estudio realizado es la posibilidad de obtener respuesta para una pregunta tan interesante como: ¿qué lenguajes son los más elegidos a la hora de desarrollar páginas web en Euskadi? La Figura 6.11 ilustra los dos gráficos necesarios para responder a esta pregunta.

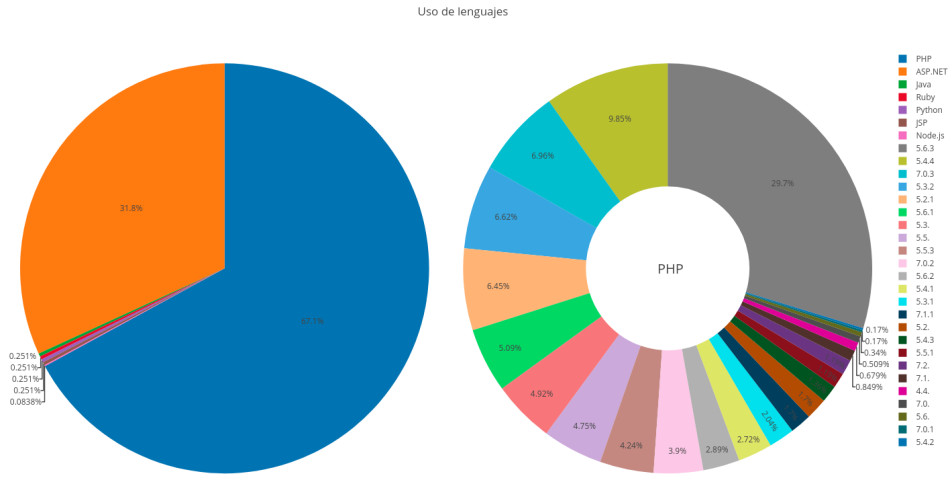


Figura 6.11: Gráfico de lenguajes de programación

Se ha podido saber, por medio del presente proyecto, que el 67 % de las páginas registradas en el sistema hacen uso de PHP como lenguaje backend. Bien es sabido que el uso de este lenguje está ampliamente extendido, por su popularidad, sencillez, soporte y por el hecho de no requerir de gran complejidad a la hora de desplegarlo en entorno productivo.

Le sigue ASP.NET, de *Microsoft*, con un uso del 32 %, tras el cual y de forma tímida podemos apreciar otros lenguajes como el omnipresente Java, al mismo tiempo que parecen comenzar a surgir alternativas comoPython, Ruby y, por último, Node.js.

Respecto a PHP, se observa que la mayoría de páginas se mantienen en la versión 5 de dicho lenguaje y que parece que pocas empresas dan el salto a la versión 7, sin alcanzar la centena de sitios web implementados partiendo de dicha versión. Esto puede ser un indicativo de que las empresas que utilizan PHP en su web, crearon ésta en algún punto de la decada que abarca de 2004 a 2014 y que no realizan un correcto mantenimiento (se ha establecido la fecha de fin de soporte a la versión 5.6 a la finalización del presente año, no prestándose este servicio, a fecha actual, para cualquier versión precedente).

6.2.4. Códigos de respuesta HTTP

También se han obtenido los resultados referentes a la respuesta devuelta por el servidor, en forma de código HTTP, al realizar la petición de recurso por parte del escaneador.

Este gráfico sirve como indicativo para poner al corriente del estado de la base de datos, en cuanto a la antigüedad de los datos almacenados. Es lógico que, al tratarse de un proyecto comenzado hace dos años, algunas de las páginas registradas hayan sufrido modificaciones en el nombre de dominio o hayan dejado de estar operativas, reflejando en otras ocasiones una respuesta de error del servidor el posible cierre de la empresa que en su día había sido identificada y registrada en el sistema. Por ello, se ha optado adicionalmente por almacenar la nueva dirección como un nuevo registro en la base de datos, en aquellos casos en que el servidor nos devuelve un código de redirección.

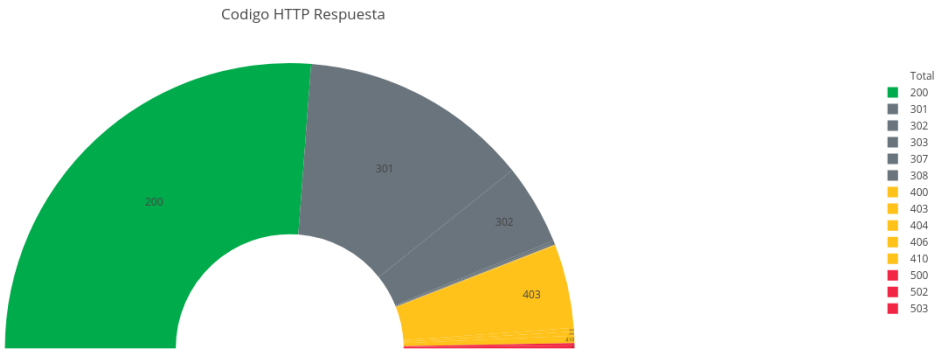


Figura 6.12: Gráfico de códigos de respuesta HTTP

Más de la mitad de las empresas almacenadas responden con un código correcto para su sitio web, denotando que la URL almacenada es la correcta y no ha sufrido modificaciones. Novecientas cuarenta han sido movidas o redireccionadas a una nueva dirección y sólo veintidos han sido permanentemente eliminadas. Trece sitios web devuelven un código referente a un error en el propio servidor, en lo que puede tratarse de servicios dados de baja por parte de la propia empresa o fruto del desuso de éstas (o, en el peor de los casos, el cierre de la empresa).

6.2.5. Servidores y versiones

El siguiente gráfico, que puede ser consultado en la Figura 6.13, muestra los diferentes software de servidor HTTP escogidos por las empresas para gestionar sus sitios web, dando el eje X la información referente a la versión concreta utilizada (cada servidor identifica visualmente todas sus versiones utilizando un mismo color), mientras que en el eje Y se indica el número de sitios web que hacen uso de dicha versión.

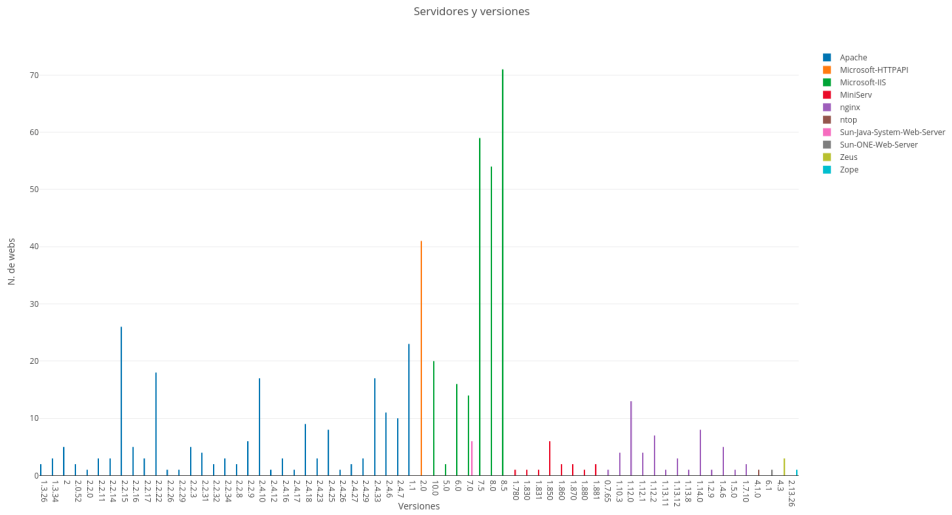


Figura 6.13: Gráfico de servidores y versiones

Los más utilizados oscilan entre *Apache*, *IIS*, de *Microsoft*, y *nginx*. En cuanto a versiones, destacan Apache 2.2.15 e IIS 8.5, en resultados que no se decantan entre el software propietario y el software libre. Esto es coherente con respecto a la sección de lenguajes de programación más utilizados, ya que PHP y ASP.NET van de la mano de estos dos servidores, respectivamente, además de mostrar una actualización similar a la de las versiones más utilizadas de estos, en cuestión de tiempo.

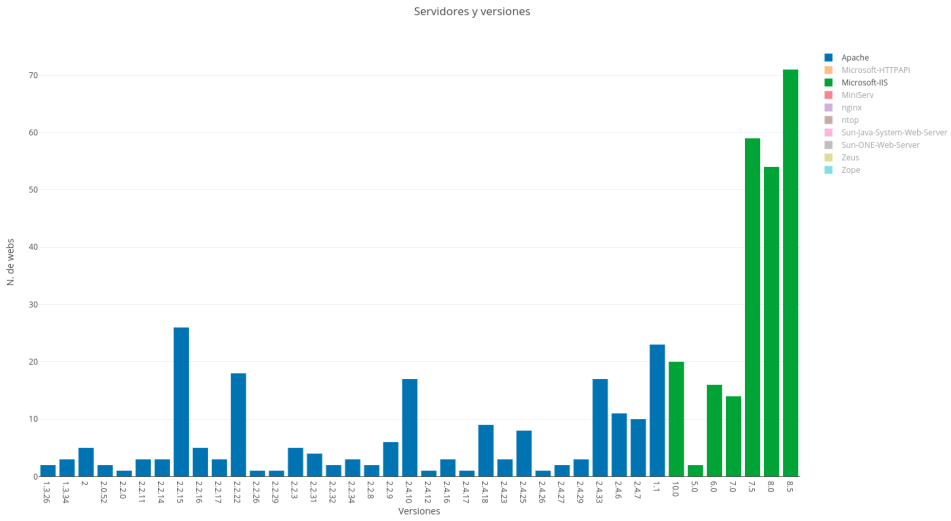


Figura 6.14: Gráfico de servidores y versiones. Apache e IIS

6.2.6. Puertos

Por último, el gráfico restante advierte de posibles agujeros de seguridad en los servidores de las distintas empresas registradas en el sistema. No se va a comentar este punto de forma extensa, aparte de especificar que lo correcto sería disponer únicamente del puerto 443 abierto, ya que este es el encargado de las conexiones HTTPS y SSL, usado para la transferencia segura y cifrada de información páginas web. Este aspecto no es conocido por las empresas, tal y como reflejan los resultados del presente estudio.

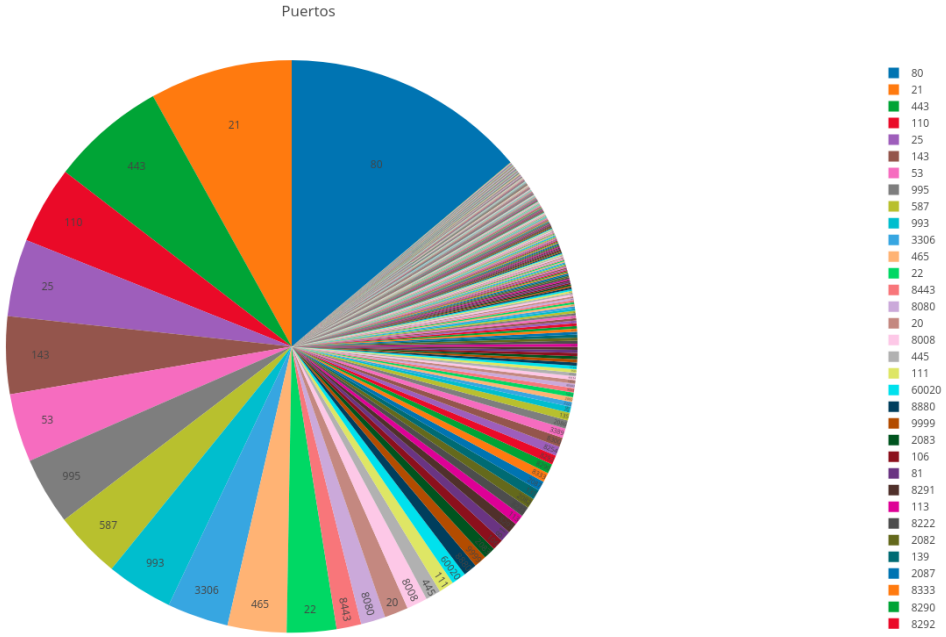


Figura 6.15: Gráfico de puertos

Principalmente, los puertos mostrados en el gráfico son utilizados para comunicaciones referentes al correo electrónico, con sus diferentes servicios, así como transferencia de recursos web, mediante HTTP o FTP. Estos resultados dejan constancia de una asignatura pendiente para muchas empresas, que no gestionan de forma correcta y eficaz los aspectos referentes a la seguridad.

6.2.7. Vulnerabilidades

Partiendo de que esta información ha de mantenerse como confidencial, siendo la propia empresa propietaria del sitio web la única que ha de tener constancia de ello, no se ha creado un gráfico exclusivo para reflejar estadísticas porcentuales de sitios web vulnerables del total de los registrados.

No obstante, cabe comentar que al concluir el escaneo global, se han obtenido un total de trescientas doce vulnerabilidades CVE diferentes, con un total de trescientos sesenta y seis posibles sitios web vulnerables, unas cifras bastante decentes, teniendo en cuenta que, para el total de empresas escaneadas, esta cifra sólo supone el 10 % de los sitios web.

En este punto cabe comentar, en referencia a los aspectos legales a tratar en el proceso de analizar las webs corporativas de empresas sin su permiso explícito, que el presente proyecto se ha realizado con fines educativos, así como que el sistema no es público aún y se va a solicitar asesoría y ayuda legal para dilucidar este aspecto, en el momento en que se decida salir a producción. Si se require permiso explícito, así se hará: el sistema no escaneará automáticamente todas las URLs del sistema, sino sólo bajo demanda de usuarios correctamente autenticados y verificados como propietarios del sitio web en cuestión.

6.3. Repositorio con resultado final

Se pueden consultar los ficheros de código fuente componentes del proyecto, en su estado final, accediendo a la dirección `https://bitbucket.org/xabicasado/xabicrawler/`.

7. Conclusiones

A lo largo de los cinco meses en los que se ha dilatado el proyecto, la dedicación e implicación en él han sido totales, comenzando por la asimilación de la finalidad de la aplicación y el aprendizaje de las tecnologías a utilizar, así como el análisis del trabajo previo (realizado en sus dos fases precedentes) y atravesando después todas las etapas de su desarrollo.

Las horas previstas para la elaboración del proyecto fueron cumplidas finalizado el segundo mes de trabajo[1]; por otra parte, el proyecto se dio por finalizado en la primera quincena del mes de julio, si bien su desarrollo había sido completado en junio y, para dicho período, se disponía de una versión de la aplicación completamente funcional y operativa. Si bien no se ha llegado a tiempo a la fecha establecida inicialmente — que marcaba la entrega del proyecto el día 28 de junio —, sí que se ha satisfecho el objetivo de finalización del proyecto en un plazo menor al semestre, logrando la cumplimentación de la totalidad de objetivos en cinco meses, como bien se ha certificado en el capítulo previo.

Este capítulo cierra la memoria del TFG *EuskalCrawler III*, que concluye los estudios académicos de Grado en Ingeniería Informática de Gestión y Sistemas de Información del alumno Xabier Casado Nieto, abordando las diferencias entre la planificación de tareas inicial y final, estudiando las posibles futuras líneas de expansión y, finalmente, realizando una valoración personal.

7.1. Análisis entre planificación estimada y real

Ninguna planificación es exacta (si bien la práctica ayuda a afinar este aspecto), más aún si a ello sumamos la aparición de riesgos que dificulten su cumplimiento. En este caso, ambos factores han sido determinantes en la fecha de finalización del proyecto, ya que, unido a la extensión en de-

[1] Con un valor de 12 créditos, a razón de 25 horas por crédito, se estima el esfuerzo horario del TFG en 300 horas

terminadas tareas debido a problemas derivados de su desarrollo, hay que añadir el hecho de que, al comienzo del mes de junio se cumplió el riesgo de mayor impacto de entre los definidos en su momento (véase: Gestión de riesgos), consistente en la contratación a jornada completa en la empresa en la que se estaban realizando las labores de becario a media jornada a lo largo de los dos últimos años. Fue preciso, debido a este contratiempo, la aplicación del plan de contingencia definido, llevando el grueso de la carga horaria semanal, en lo que a jornada de trabajo se refiere, al fin de semana, si bien fue de agradecer que las labores de desarrollo, referentes al segundo y tercer módulo del EDT, habían finalizado para dicha fecha.

Por otra parte, cabe destacar que no se ha considerado precisa la inclusión de tarea adicional alguna, más allá de las inicialmente identificadas y definidas (véase: Alcance), modificando únicamente la tarea de diseño de diagramas de secuencia por diagramas de actividades.

Con todo lo anterior comentado, se exponen las tablas finales con las duraciones finales de cada tarea.

ID	***Nombre de la tarea***	***Duración estimada***	***Duración final***
	EuskalCrawler III	490 horas	720 horas
1	***Organización y aprendizaje***	138 horas	242 horas
1.1	Planificación de las tareas	8 horas	12 horas
1.2	Configuración de la máquina de desarrollo	2 horas	2 horas
1.3	Creación y configuración del repositorio	2 horas	2 horas
1.4	Aprendizaje de Django	72 horas	40 horas
1.5	Análisis e interpretación del código actual	20 horas	108 horas
1.6	Documento de Objetivos de Proyecto	34 horas	78 horas
1.6.1	Elección de herramientas	2 horas	2 horas
1.6.2	Diseño de diagramas	4 horas	4 horas
1.6.3	Redacción del DOP	28 horas	72 horas

Tabla 7.1: Análisis de planificación estimada y real del módulo de Organización y aprendizaje

ID	*Nombre de la tarea*	*Duración estimada*	*Duración final*
2	***Captura, almacenamiento y análisis de resultados***	214 horas	258 horas
2.1	Captura de requisitos	12 horas	7 horas
2.1.1	Especificación de Requisitos de Software	8 horas	4 horas
2.1.2	Diagrama de casos de uso	2 horas	1 hora
2.1.3	Modelo de dominio	2 horas	2 horas
2.2	Análisis y diseño	18 horas	21 horas
2.2.1	Rediseño del Diagrama de Entidad-Relación	8 horas	16 horas
2.2.2	Diseño del diagrama de clases	2 horas	1 horas
2.2.3	Diseño del diagrama de actividades	8 horas	4 horas
2.3	Implementación	168 horas	214 horas
2.3.1	Implementación del modelo	4 horas	8 horas
2.3.1.1	Población de la base de datos	4 horas	4 horas
2.3.2	Escaneadores	60 horas	102 horas
2.3.2.1	Elección de las API a utilizar	20 horas	30 horas
2.3.2.2	Implementación de los métodos de escaneo	40 horas	72 horas
2.3.3	Implementación de las vistas	72 horas	72 horas
2.3.4	Implementación del script de almacenamieto	8 horas	12 horas
2.3.5	Implementación del script de análisis de resultados	20 horas	16 horas
2.4	Diseño de las pruebas	16 horas	16 horas

Tabla 7.2: Análisis de planificación estimada y real del módulo de Captura, almacenamiento y análisis de resultados

ID	*Nombre de la tarea*	*Duración estimada*	*Duración final*
3	***Interfaz de usuario web***	58 horas	62 horas
3.1	Captura de requisitos	10 horas	6 horas
3.1.1	Especificación de Requisitos de Software	4 horas	2 horas
3.1.2	Diseño de los casos de uso	4 horas	2 horas
3.1.3	Diseño del modelo de dominio	2 horas	2 horas
3.2	Implementación	48 horas	56 horas
3.2.1	Implementación de las plantillas	8 horas	12 horas
3.2.2	Rediseño de las hojas de estilos	40 horas	44 horas
3.2.2.1	Prototipado de la IU web	4 horas	8 horas
3.2.2.2	Implementación de la IU web	36 horas	36 horas
4	***Implantación***	42 horas	42 horas
4.1	Despliegue	2 horas	2 horas
4.2	Pruebas	4 horas	8 horas
4.3	Ejecución de los scripts	36 horas	32 horas
5	***Documentación***	38 horas	116 horas
5.1	Manual para el despliegue en el servidor	2 horas	4 horas
5.2	Manual de usuario	4 horas	4 horas
5.3	Redacción de la memoria	32 horas	108 horas

Tabla 7.3: Análisis de planificación estimada y real de los módulos Interfaz de usuario web, Implantación y Documentación

En base a los tiempos reales, el cómputo total asciende, de las 490 horas inicialmente estimadas, a un total de 720 horas finales de dedicación al proyecto, siendo necesaria la modificación de la fecha de entrega, pasando de la convocatoria de junio — marcada inicialmente en los objetivos — a la convocatoria de julio, a fin de entregar un producto finalizado y de calidad, mayormente en lo que a documentación se refiere.

Llegados a este punto, también se considera oportuno calcular el coste total acumulado del proyecto, derivado de la suma de la evaluación económica de la fase actual a la obtenida en las dos anteriores fases, a fin de valorar el esfuerzo y gasto en equipo preciso para alcanzar el estado actual. No se va a realizar una reevaluación económica, puesto que carece de sentido modificar el valor a un proyecto por cuestiones de una planificación inicial que difiere del esfuerzo final real.

Dicho esto, en la Tabla 7.4 se muestran los costes totales de cada una de las fases, así como el coste final acumulado.

Versiones	***Coste (en €)***
Euskal Crawler	5797.65 €
EuskalCrawler, un detector de vulnerabilidades de sitios web	4140.06 €
EuskalCrawler III	5497.82 €
Total	15435.53 €

Tabla 7.4: Coste final de EuskalCrawler

El valor del proyecto asciende finalmente a los 15435.53 €, llegados a la finalización de su tercera fase.

7.2. Futuras líneas de desarrollo

Debido al principal *handicap* del proyecto, el cual no es otro que el plazo de entrega y reducido tiempo disponible para su elaboración, comentado previamente en la introducción del presente capítulo, se han descartado — bien en el planteamiento de objetivos iniciales, bien según estas ideas surgían a lo largo de la fase de desarrollo — varios puntos de notorio interés y que pueden ser tenidos en cuenta a la hora de servir como base para una posible continuación a la presente etapa, en forma de un nuevo TFG. A continuación, se exponen dichas funcionalidades:

- **Migración de la base de datos a un modelo no-relacional**, en concreto a un sistema de base de datos orientado a documentos, como puede ser el caso de *MongoDB*. Este punto, tal como se ha comentado previamente (véase: Base de datos), implicaría una mejora en el almacenamiento de resultados, al estandarizar el formato de gestión de los datos, por parte tanto de las diferentes APIs involucradas en el proceso de captura de información, como de la capa de persistencia. Para su migración, sería conveniente un estudio previo de ventajas e inconvenientes derivados del hecho de dejar de trabajar con un sistema de base de datos orientado a entidades y relaciones, que destaca por la garantía en la integridad y no redundancia de datos, en pro de un sistema que registraría fielmente los resultados de las diferentes herramientas utilizadas en el proyecto.

- **Llamadas asíncronas al servidor**, a fin de evitar la tediosa y frustrante recarga de páginas. Su ausencia en el proyecto se debe a dos razones: por una parte, que la inclusión de la tecnología *Asynchronous JavaScript And XML (AJAX)* requiere modificar completamente la forma de comunicación entre plantillas y vistas respecto al estándar de *Django* (implicando, entre otros, la implementación de validaciones en formularios y paso de parámetros del controlador) y, por otra parte, el cumplimiento de los Requisitos de Interfaces Externos, que establecen que la web ha de ser totalmente funcional para cualquier dispositivo y navegador enumerado en éstos (véase: Requisitos específicos), lo cual dejaría de cumplimentarse para cualquier navegador en el que *JavaScript* se encontrase desactivado.

- **Creación de una API** de *EuskalCrawler*, a fin de que el proyecto pudiera ser reutilizable en otras aplicaciones de forma sencilla, proveyendo a la comunidad de desarrolladores una serie de métodos para dicho fin. Este punto requiere que el proyecto disponga de una URL pública en entorno productivo; si bien actualmente se satisface el primer punto, no se puede considerar que ésta se encuentre disponible en un entorno productivo, puesto que requiere de un despliegue al uso y configuración de un servidor DNS.

- **Panel de administración** con posibilidad de consulta de gráficos a medida, en los que mostrar información confidencial a la que sólo personal autorizado tenga acceso, a saber: páginas con vulnerabilidades detectadas, opciones de filtrado por zonas/provincias, estadísticas de diferentes usuarios empresarios, posibilidad de desactivación de usuarios (actualmente disponible, accediendo directamente al sistema de gestión de base de datos) y un sinfín de combinaciones que pueden ser planteadas y estudiadas en el momento de la inclusión de este punto.

- **Utilización de la API de Vulners.com**. Escaneador que finalmente se quedó en el tintero, ya que su funcionamiento difiere con respecto del resto de las herramientas utilizadas y su inclusión conllevaría a un rediseño del algoritmo de escaneo implementado. Para su utilización, sería necesario el envío en entrada de cada uno de los resultados individuales obtenidos por cada uno de los escaneadores, para el cual éste devolvería en salida si presentan una vulnerabilidad en el sitio web escaneado. El incremento en el tiempo de escaneo sería notable, por lo que su uso ha sido descartado, si bien complementaría a los actuales resultados de *shodan.io*, permitiendo certificar si una vulnerabilidad es tal.

- **Extraer las funciones de escaneo** a ficheros independientes, para facilitar la modularidad y ampliación de escaneadores a futuro, sin necesidad de editar el fichero de vistas existente (si bien actualmente, tras simplemente realizar la edición del fichero comentado, la operatividad del nuevo escaneador es completa).
- **Aviso de vulnerabilidades a empresas**, realizado mediante el envío de un correo, bien a la dirección de la propia página detectada en el escaneo o a la utilizada en el registro del usuario propietario de la empresa. Sería interesante que, tras la finalización del escaneo global que se realiza mensualmente, se avisara a los propietarios de las páginas con vulnerabilidades de dicha situación, para ponerles al corriente y tratar de facilitar su inmediata resolución.

7.3. Valoración personal

Nuevamente, se recurre a la utilización de la primera persona para la redacción de esta reflexión final, afirmando en primer lugar que es innegable el sentimiento de satisfacción una vez concluido el plazo de entrega, que inicialmente acogí con ilusión y entusiasmo, y seguidamente continué con expectativa.

A modo de retrospección y como bien se comenta en la introducción de esta sección, el presente Trabajo de Fin de Grado concluye mis estudios académicos, en lo que ha supuesto un proceso de un total de nueve años, desde que finalizados mis estudios obligatorios, decidí que mi futuro pasaba por la informática y decidí dar el salto a través de la Formación Profesional, comenzando por cursar un Grado Medio, para continuar con dos Grados Superiores y optar finalmente por el acceso a la universidad, en el presente Grado en Ingeniería Informática de Gestión y Sistemas de Información.

Un punto a destacar es que este proyecto me ha permitido aprender tanto un lenguaje como un framework (Python y Django, respectivamente) con los que, hasta el inicio del Trabajo de Fin de Grado, nunca había trabajado y cuyo uso está cada vez más extendido y demandado. El simple hecho de haber logrado realizar un sistema funcional y completo, dominando y utilizando estas tecnologías, ya supone una victorial moral.

Puedo concluir con certeza que los pasos seguidos hasta llegar al resultado definitivo han resultado fundamentales en mi formación, permitiéndome aglutinar — y, a su vez, ampliar — todos los conocimientos adquiridos a

lo largo de estos nueve años para materializarlos en el presente trabajo, el cual ha supuesto al mismo tiempo mi primer proyecto de gran envergadura realizado de forma individual, no sin destacar la participación del director de proyecto, Juanan, quien ha tutelado en todo momento el progreso de *EuskalCrawler*, aportando siempre un punto de vista profesional en el trabajo y organización del mismo.

Anexos

Anexo I: Manual para despliegue de EuskalCrawler en servidor

Se asume que el despliegue de la aplicación se va a realizar en un servidor Microsoft Azure con instalación limpia de Ubuntu Server (en concreto, se ha realizado para su versión 17.10). Con esta premisa, el presente manual no contemplará posibles errores durante el proceso si se realiza en un entorno diferente al comentado. Se accederá de forma remota al servidor, mediante terminal y utilizando una conexión SSH.

```
ssh xabier@40.89.128.211
```

El primer paso — sin necesidad aún de conectarse remotamente — es abrir el puerto 8000 del servidor desde la web de configuración del proveedor.

A.1. Instalación Python

El siguiente paso consiste en la instalación de Python, previo a lo cual (y a fin de evitar posteriores errores) estableceremos que el lenguaje por defecto del servidor es el español. Para ello, introduciremos las siguientes órdenes:

```
export LANG=es_ES.UTF-8

sudo locale-gen es_ES.UTF-8

sudo dpkg-reconfigure locales

sudo dpkg-reconfigure tzdata
```

Tras ello, ahora sí procederemos a instalar Python (en concreto, en su versión 2.7), así como su gestor de paquetes *pip*, el cual nos será de gran utilidad a la hora de instalar las dependencias requeridas por el proyecto:

```
sudo apt-get install python 2.7

sudo apt-get install -y make build-essential
libssl-dev zlib1g-dev libbz2-dev libreadline-dev
libsqlite3-dev

wget curl llvm

apt install python-pip

pip install pillow
```

En ocasiones, un bug que afecta a la versión 10 de pip nos obliga a utilizar el comando alternativo:

```
python -m pip install nombre-del-paquete
```

A.2. Instalación Django

Para este paso, basta con introducir el siguiente comando:

```
pip install -U django===1.9.5
```

A.3. Instalación MySQL

En primer lugar, instalamos el servidor MySQL, mediante los dos siguientes comandos (seguimos las instrucciones del asistente en el segundo paso):

```
sudo apt-get install mysql-server

mysql_secure_installation mysql_secure_installation
```

Una vez finalizada la instalación accedemos a MySQL y definimos la nueva base de datos, así con permisos sobre ella:

```
CREATE DATABASE euskalcrawler;

CREATE USER 'xabier'@'localhost' IDENTIFIED BY
'contraseña';

GRANT ALL PRIVILEGES ON euskalcrawler.* TO
'xabier'@'localhost' WITH GRANT OPTION;

GRANT ALL PRIVILEGES ON test_euskalcrawler.* TO
'xabier'@'localhost' WITH GRANT OPTION;
```

Si bien no es obligatorio, sí resulta conveniente instalar, adicionalmente, las siguientes librerías de compatibilidad entre Python y MySQL:

```
sudo apt-get install libmysqlclient-dev

sudo apt-get install python-mysqldb
```

A.4. Instalación módulos y dependecias necesarias

Procederemos a instalar los módulos de Python necesarios para la correcta ejecución del proyecto:

```
sudo apt-get install ruby ruby-dev libcurl4-gnutls-dev
libxml2 libxml2-dev libxslt1-dev

sudo gem install json

gem install bson

gem install bson_ext

gem install mongo

gem install rchardet
```

Tras este paso, tenemos que editar un fichero de configuración, a fin de solucionar un bug de *Ruby*:

```
nano -c /usr/lib/ruby/vendor_ruby/rchardet/
universaldetector.rb
```

Insertamos en la primera línea el siguiente comentario:

```
# encoding: US-ASCII
```

Procedemos a continuar con la instalación de las dependencias:

```
apt-get install whatweb

pip install shodan

apt-get install nmap

pip install python-nmap

pip install plotly

pip install sendgrid

pip install requests

pip install django-widget-tweaks
```

A.5. Instalación git y descarga del repositorio

Una vez instaladas todas las dependencias y configurado el entorno productivo, es hora de descargar el repositorio del proyecto. Para ello, comenzamos configurando la herramienta git, previa descarga:

```
apt-get install git

git config --global user.name "Xabier Casado"

git config --global user.email "xabicasadonieto@gmail.com"
```

Creamos una clave SSH mediante el comando:

```
ssh-keygen
```

La asociamos desde nuestra cuenta de Bitbucket y probamos que funciona correctamente con el siguiente comando

```
ssh -T git@bitbucket.org
```

Por último, clonamos el repositorio en la ubicación deseada del servidor:
`git clone git@bitbucket.org:xabicasado/xabicrawler.git euskalcrawlerProject`

Hay que modificar el archivo setting.template — indicando: la clave única de la aplicación (`SECRET_KEY`), los parámetros de conexión a la base de datos (`DATABASES`), la dirección IP del servidor (`SERVER_IP`), las claves personales de las APIs utilizadas (`SENDGRID_API_KEY`, `PLOTLY_USERNAME`, `PLOTLY_API_KEY`, `SHODAN_API_KEY` y `WHATCMS_API_KEY`), así como el correo con el que enviar los correos de confirmación de registro (`DEFAULT_FROM_EMAIL`) —, y renombrarlo como settings.py

A.6. Creación del usuario administrador de la aplicación

Desde el directorio raíz del proyecto, procedemos en primer lugar a migrar la base de datos del proyecto, así como las tablas que Django requiere para la gestión de usuarios y sesiones de un proyecto implementado en dicho framework:

```
python manage.py makemigrations euskalcrawler
```

```
python manage.py migrate
```

Tras ello, introducimos la orden necesaria para la creación del administrador:

```
python manage.py createsuperuser
```

A.7. Carga de datos iniciales en base de datos

Para completar esta tarea, tenemos que ejecutar el script encargado de insertar la información proveniente de las versiones anteriores, así como la

necesaria para la correcta configuración inicial de la aplicación:

```
python populate_euskalcrawler.py
```

Nota: las tablas Empresa, Escaneador y Propietarios han sido añadidas en el Panel de Administración del proyecto Django, a fin de proporcionar una interfaz gráfica que facilite el mantenimiento de éstas por parte del usuario creado en el paso anterior.

A.8. Despliegue local de la aplicación

Introduciendo el siguiente comando, permitimos el despliegue de la aplicación en segundo plano, lo que nos da la opción de cerrar la terminal y que la aplicación siga en línea:

```
nohup python manage.py runserver 0.0.0.0:8000 &
```

A.9. Automatización de procesos mediante Crontab

En primer lugar, introducimos el comando:

```
crontab -e
```

Al ejecutar la anterior orden, se abre en modo de edición el fichero de tareas *cron*, en el cual pegamos las siguientes tres líneas:

```
0 0 1 * * /usr/bin/python /home/xabier/
euskalcrawlerProject/escaneo_empresas.py
> /home/xabier/euskalcrawlerProject/results/log_output
2> /home/xabier/euskalcrawlerProject/results/error_output

0 0 * * 0 /usr/bin/python /home/xabier/
euskalcrawlerProject/actualiza_graficos.py

0 0 * * * mysqldump --defaults-file=/home/xabier/
.credenciales.cnf euskalcrawler> /home/xabier/
bd_backups/euskalcrawler-$(date '+%Y%m%d').sql
```

A.10. Testeo de la aplicación

Podemos, finalmente, comprobar que el proceso ha finalizado satisfactoriamente, mediante la ejecución de los tests de la aplicación, tal que:

```
python manage.py test euskalcrawler
```

Anexo II: Manual de usuario de EuskalCrawler

El presente manual está pensado como una guía de referencia para la fácil comprensión de la aplicación. En él se encuentran recogidas todas las posibles operaciones a realizar desde el sitio web desarrollado a lo largo del TFG.

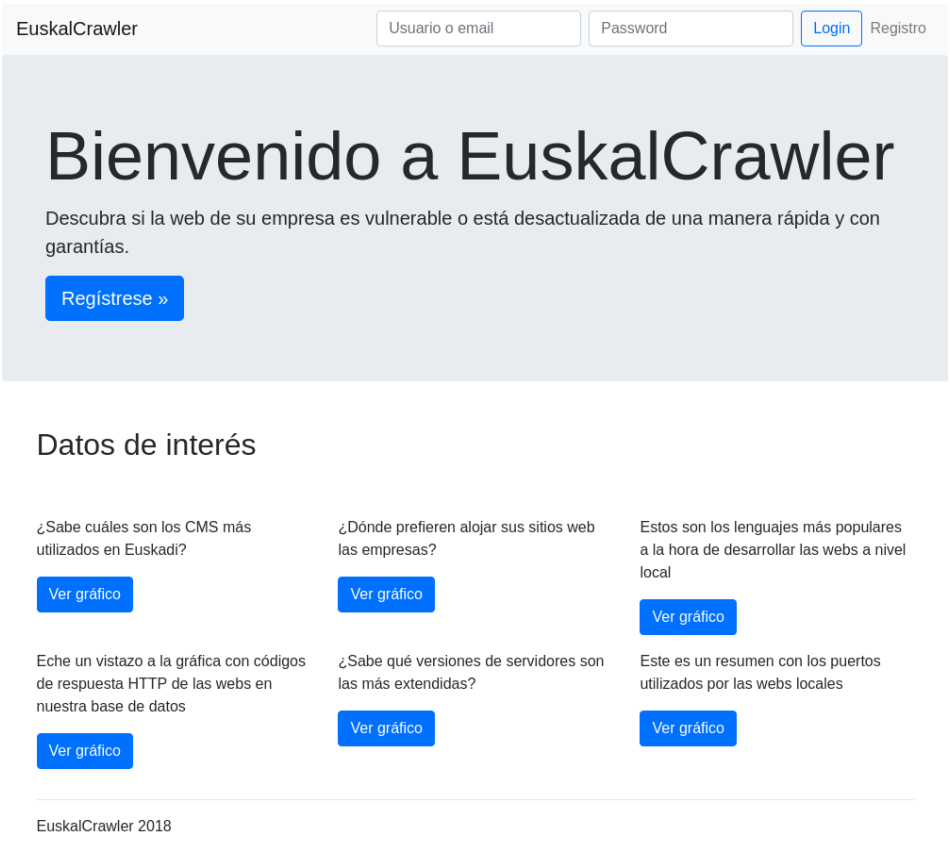


Figura B.1: Página principal de EuskalCrawler

Para acceder a *EuskalCrawler* basta con introducir la siguiente URL en el navegador:

`http://euskalcrawler.francecentral.cloudapp.azure.com:8000/`

Tras acceder, se mostrará la página principal, tal y como se aprecia en la Figura B.1. Desde ella, podemos identificarnos con nuestro usuario o email y contraseña, acceder a la página de registro o visualizar las gráficas de *Datos de interés*. El registro en la página web es condición indispensable para tener acceso a la herramienta de escaneo.

B.1. Registro

Para comenzar el proceso de por el cual darse de alta en la aplicación, basta con acceder a la página de registro, pulsando bien el enlace *Registro* en la parte derecha de la barra superior de navegación o el botón *Regístrese* disponible en la página principal.

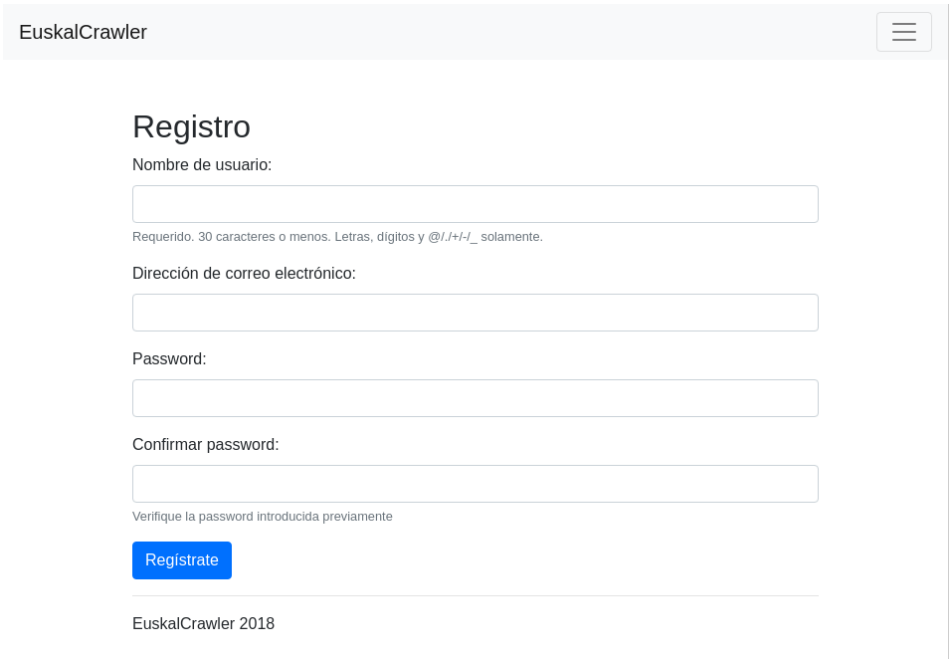


Figura B.2: Formulario de registro

En esta página es obligatorio informar todos los campos a fin de realizar

el registro del nuevo usuario. En caso de que algún campo no cumpla con las especificaciones, se mostrará el correspondiente mensaje y se volverá a solicitar al usuario.

Una vez completado el registro, se enviará un mensaje a la dirección de correo electrónico indicada, con un enlace de confirmación, tras la cual se habilitará la nueva cuenta, permitiendo la identificación y acceso del usuario a la aplicación.

B.2. Identificación

Para el acceso a la aplicación, previo registro de una cuenta, es necesario ingresar nombre de usuario o email y contraseña en los campos habilitados para tal fin, disponibles en la barra superior de navegación.

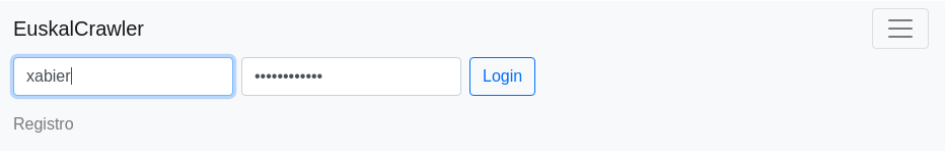


Figura B.3: Formulario de registro

Si los datos son correctos, se redirigirá al usuario a su página personal; en caso contrario, se indicarán los detalles de la identificación errónea.

B.3. Consulta de datos de interés

Se permite acceder a las gráficas de resultados desde la página principal, sin necesidad de estar autenticado. No obstante, una vez logueados, podemos pulsar el enlace *Datos de interés*, ubicado en la barra superior de navegación.

Todos los gráficos son interactivos y se permite la selección de éstos pulsando el botón *Ver gráfico* bajo la frase descriptiva de cada uno de ellos.

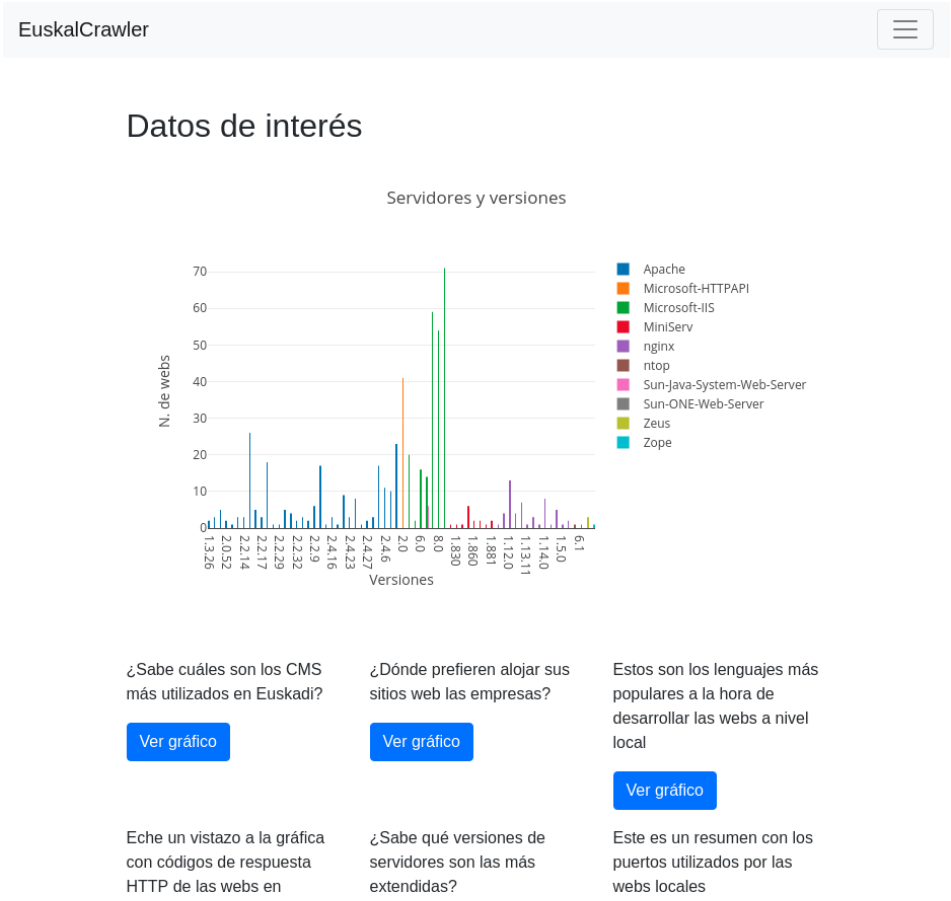


Figura B.4: Consulta del gráfico relativo a servidores

Para una experiencia completa — si bien los gráficos están realizados atendiendo a un diseño responsivo — se recomienda su visualización desde una resolución superior a los 768 píxeles de ancho.

B.4. Realizar un escaneo

Este proceso se desglosa en tres operaciones, las cuales se describen a continuación.

B.4.1. Registro del sitio web

El primer paso a realizar para el escaneo de un sitio web es, lógicamente, el registro de la propia web.

Para dicha operación, se ha habilitado un formulario en la parte superior de la página personal del usuario autenticado, de acuerdo a la Figura B.5.

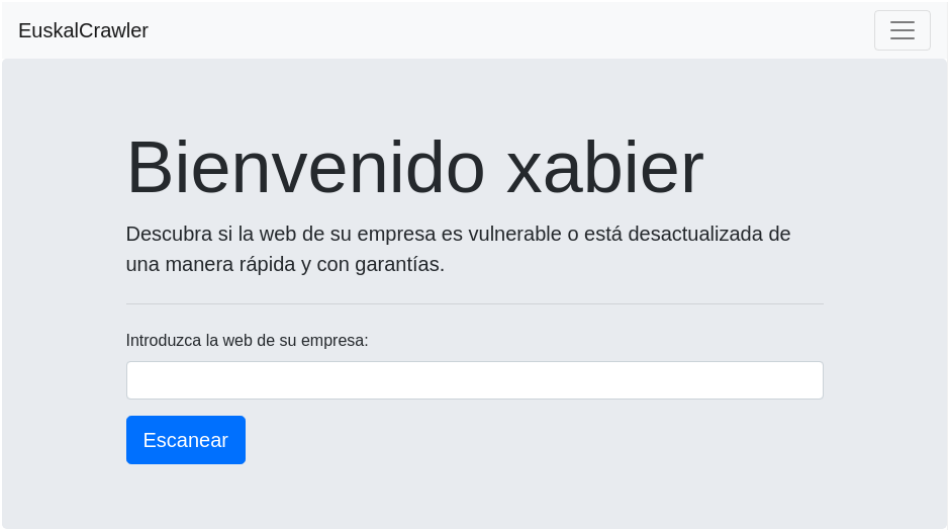


Figura B.5: Formulario para el registro de una web corporativa

Una vez introducida una URL correcta, se mostrará junto al resto de sitios web registrados por el usuario en su página personal.

B.4.2. Verificación del sitio web

Para proceder al escaneo del sitio web previamente registrado hay que demostrar la autoría de éste, lo cual la aplicación verifica solicitando al usuario la creación de contenido dentro del propio sitio web. Este proceso se realiza de forma externa a la aplicación, que únicamente se mantendrá a la espera de que el usuario realice la operación indicada, facilitando a éste un código identificador único, que será el nombre que el mismo tendrá que dar al fichero creado y, al mismo tiempo, también el contenido de dicho de fichero.

Figura B.6: Web en proceso de verificación

B.4.3. Escaneo del sitio web

Por último, una vez satisfecho el requisito anterior, se mostrará un botón *Escanear* para el sitio web verificado.

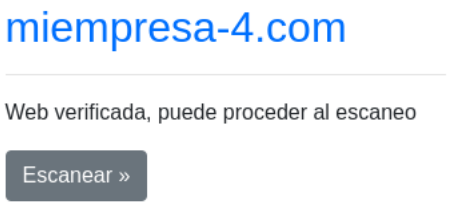


Figura B.7: Web verificada y preparada para su escaneo

Una vez escaneado un sitio web, se muestra la fecha de éste y siempre se permite realizar un nuevo escaneo, mediante el enlace *Volver a escanear*.

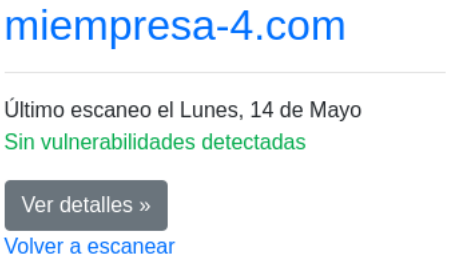


Figura B.8: Volver a escanear

B.5. Consulta de resultados de un escaneo

Desde la página principal tendremos acceso a todos los sitios web de empresas que hayamos dado de alta previamente y podremos visualizar, adicionalmente, el estado en que se encuentra cada uno de ellos, pudiendo ser:

- Pendiente de verificación
- Pendiente de escaneo
- Escaneado sin vulnerabilidades detectadas
- Escaneado con vulnerabilidades detectadas

La Figura B.9 ilustra la página personal de un usuario con cuatro páginas registrada, cada una de ellas con uno de los anteriores estados.

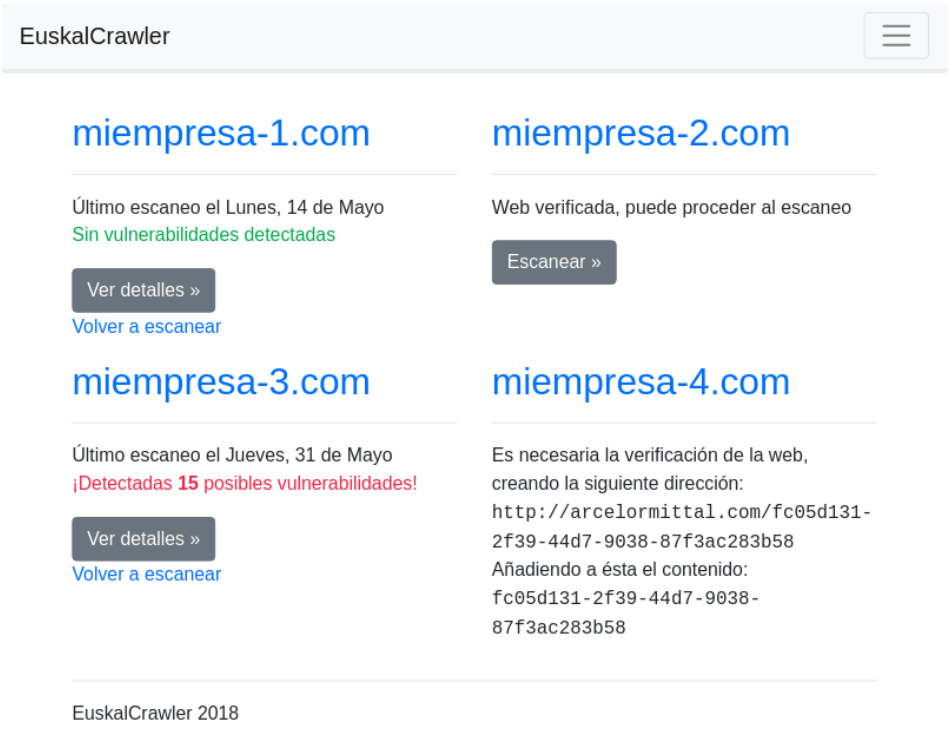


Figura B.9: Página personal del usuario

Para cualquiera de los dos últimos estados, se nos permite acceder a la página de resultados, en la cual se mostrará la información relevante del escaneo, así como la lista de vulnerabilidades, en caso de haberlas.

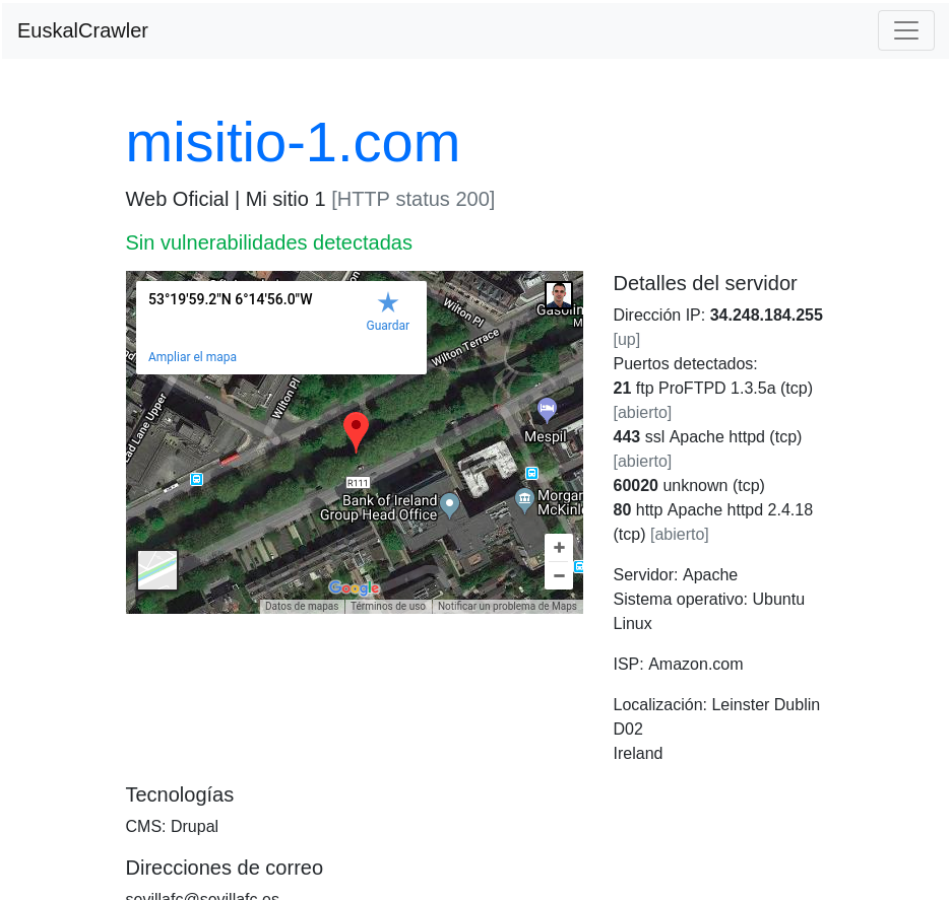


Figura B.10: Página de resultados

B.6. Desconexión

Para cerrar sesión, simplemente es necesario pulsar el enlace Cerrar sesión, ubicado en la esquina derecha de la barra superior de navegación.

Anexo III: Portada oficial del TFG firmada

Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

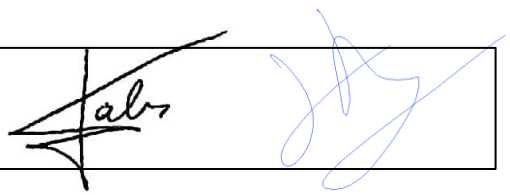
GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

EUSKALCRAWLER III

Alumno: Casado Nieto, Xabier

Director: Pereira Varela, Juan Antonio



Curso: 2017-2018

Fecha: Bilbao, 17 de julio de 2018

Acrónimos

AJAX Asynchronous JavaScript And XML. 112

API Application Programming Interface. 7, 10, 12, 26, 37, 71, 73, 89, 109, 111, 112, 121, *Glosario de términos:* interfaz de programación de aplicaciones

CAV Comunidad Autónoma Vasca. 3

CIDAN Confidencialidad, Integridad, Disponibilidad, Autenticación y No-repudio. 2

CMS Content Management System. 2, 73, 97–99, *Glosario de términos:* sistema de gestión de contenidos

CSS Cascading Style Sheets. 12, 76, *Glosario de términos:* hoja de estilo en cascada

CVE Common Vulnerabilities and Exposures. 2, 61, 73, 106

DER Diagrama de Entidad-Relación. 23–25, 37, 60, 63, 64, 75, 109

DML Data Manipulation Language. 74, *Glosario de términos:* lenguaje de manipulación de datos

DOP Documento de Objetivos de Proyecto. 9, 16, 19, 20, 22, 30, 36, 48, 108

EDT Estructura de Descomposición del Trabajo. 15, 17, 19–21, 29, 33, 34, 67, 108

ERS Especificación de Requisitos de Software. 6, 22, 23, 30–32, 37, 38, 53, 60, 79, 109, 110

HTML HyperText Markup Language. 12, 65, 76, *Glosario de términos:* html

HTTP Hyper Text Transfer Protocol. 72, 102, 103, *Glosario de términos:* protocolo de transferencia de hipertexto

IDE Integrated Development Environment. 18, 71, *Glosario de términos:* entorno de desarrollo integrado

ISP Internet Service Provider. 73, 100

IU interfaz de usuario. 3, 16, 29, 31, 32, 38, 57, 110

JSON JavaScript Object Notation. 71–73, 75, *Glosario de términos:* json

LSI Lenguajes y Sistemas Informáticos. 2

MTV Model-Template-View. 13

MVC Modelo-Vista-Controlador. 13, 14

OMG Object Management Group. 140

ORM Object-Relational mapping. 74

PHP Personal Home Page Hypertext Preprocessor. 3, 101, 103, *Glosario de términos:* php

RF Requisito Funcional. 53–57, 80, 83, 84, 86, 88–92

RIE Requisito de Interfaz Externo. 53, 57, 76, 92, 95, 112

SSH Secure SHell. 117, *Glosario de términos:* intérprete de órdenes seguro

TFG Trabajo de Fin de Grado. 2, 5, 7, 9, 17, 19, 34, 35, 39, 48, 107, 111, 113, 125

TIC Tecnologías de la Información y la Comunicación. 1

UML Unified Modeling Language. 12, *Glosario de términos:* lenguaje unificado de modelación

URL Uniform Resource Locator. 3, 25, 56, 61, 69, 72, 102, 106, 112, 126, 129, *Glosario de términos:* localizador de recursos uniforme

Glosario de términos

antipatrón de diseño Es un patrón de diseño que invariablemente conduce a una mala solución para un problema. 40

backend Parte del desarrollo web que se ejecuta en el servidor, encargada de la manipulación de los datos introducidos por el usuario en el frontend, interacción con la base de datos y generación de plantillas. 3, 16, 20, 22, 23, 25, 33, 101

crawling Técnica de rastreo de webs automatizado y recursivo cuyo fin es identificar hipervínculos e indexar contenido. 3

deadline Anglicismo utilizado con frecuencia en todo tipo de sectores empresariales. El término se puede traducir como plazo de entrega o fecha límite. 9, 45

diseño responsivo Técnica de diseño web que busca la correcta visualización de una misma página en distintos dispositivos, desde ordenadores de escritorio a tablets y móviles. 6, 10, 28, 32, 57, 76, 95, 128

django Framework para aplicaciones web gratuito y open-source, escrito en Python. 3, 5, 7, 13, 14, 18, 25, 36, 63–65, 67, 74, 79, 108, 112, 113, 121

entorno de desarrollo integrado Aplicación informática que proporciona servicios integrales a fin de facilitar al programador el desarrollo de software. Normalmente consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de los IDE tienen auto-completado inteligente de código. 12, 17–19, 25–28, 31, 32, 138

framework Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar. 3, 5, 13, 18, 65, 71, 74, 76, 113, 121

frontend Parte del desarrollo web ejecutada en el cliente, responsable de recoger los datos de entrada del usuario y enviarlos al servidor para su procesamiento, previa validación. Adicionalmente, se encarga de recibir y exponer al usuario, los mensajes y datos recibidos desde el backend, en un lenguaje entendible por éste. 16, 28, 30, 31, 33, 76, 95

hoja de estilo en cascada Lenguaje de diseño gráfico cuyo fin es definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Su función es separar la estructura de la presentación. 32, 76, 95, 137

html Lenguaje de marcado utilizado para el desarrollo de páginas web, sirviendo para definir su estructura y contenido. 137

interfaz de programación de aplicaciones Conjunto de reglas y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas. 7, 25, 26, 53, 137

intérprete de órdenes seguro Protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse y controlar un host remotamente, encriptando todo lo que se envía y recibe. 138

json formato de texto ligero para el intercambio de datos basado en documentos, estándar para representar datos estructurados en la sintaxis de objetos de JavaScript. Es comúnmente utilizado para transmitir datos en aplicaciones web. 138

lenguaje de manipulación de datos Lenguaje proporcionado por los sistemas gestores de bases de datos que permite a los usuarios de la misma llevar a cabo las tareas de consulta o modificación de los datos (sentencias SELECT, INSERT, UPDATE y DELETE). 137

lenguaje de marcado Forma de codificar un documento que, junto con el texto, incorpora etiquetas o marcas que contienen información adicional acerca de la estructura del texto o su presentación. 140

lenguaje unificado de modelación Lenguaje gráfico destinado al modelado de sistemas y procesos, basado en la orientación a objetos y promovido por el Object Management Group (OMG). Está unificado, ya que deriva de notaciones precedentes [Debrauwer y Van der Heyde, 2001]. 138

localizador de recursos uniforme Dirección específica, definida mediante una secuencia de caracteres que sigue un estándar y que se asigna a cada uno de los recursos disponibles en una red (p. ej., Internet), con la finalidad de que puedan ser localizados o identificados. 138

php Lenguaje backend de código abierto, especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. 138

protocolo de transferencia de hipertexto método de comunicación de uso más extendido en Internet y que permite el intercambio de información entre los diferentes servicios y clientes que utilizan una página web. 137

python Lenguaje de programación multiparadigma, que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. 3, 7, 11, 12, 18, 65, 68, 74, 76, 101, 113, 117–119

scraping Técnica de rastreo de webs automatizado cuyo fin es extraer información de forma estructurada. 3, 6

sistema de gestión de contenidos Programa informático que permite crear una estructura de soporte framework para la creación y administración de contenidos, principalmente en páginas web. 2, 97, 137

tethering Proceso por el cual un dispositivo móvil con conexión a Internet actúa como pasarela para ofrecer acceso a la red a otros dispositivos, cualesquiera que estos sean, asumiendo dicho dispositivo móvil un papel similar al de un router inalámbrico. 44

Bibliografía

[Azzopardi y Maxwell, 2016] Azzopardi, L. y Maxwell, D. (2016). *Tango with Django*. tangowithdjango.com.

[Brown *et al.*, 1998] Brown, W., Malveau, R., McCormick, H., y Mowbray, T. (1998). *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*. Robert Ipsen.

[Costas, 2011] Costas, J. (2011). *Seguridad y alta disponibilidad. CFGS*. RA-MA Editorial.

[Debrauwer y Van der Heyde, 2001] Debrauwer, L. y Van der Heyde, F. (2001). *UML 2. Iniciación, ejemplos y ejercicios corregidos*. Editions ENI.

[Gallego, 2018] Gallego, I. (2018). Euskalcrawler, un detector de vulnerabilidades de sitios web. Disponible en: `https://addi.ehu.es/handle/10810/25787`.

[Holovaty y Kaplan-Moss, 2009] Holovaty, A. y Kaplan-Moss, J. (2009). *The Definitive Guide to Django. Web Development Done Right*. Apress.

[Instituto Vasco de Estadística, 2017] Instituto Vasco de Estadística (2017). Equipamientos de tecnologías de la información en los establecimientos de la c.a. de euskadi por territorio histórico, rama de actividad (a38) y estrato de empleo (%). Disponible en: `http://www.eustat.eus/elementos/tbl0000713_c.html`.

[Instituto Vasco de Estadística, 2018] Instituto Vasco de Estadística (2018). Cuatro de cada cinco personas son usuarias de internet en la c.a. de euskadi en 2018. Disponible en: `http://www.eustat.eus/elementos/not0015229_c.html`.

[López, 2016] López, D. (2016). Euskal crawler. Disponible en: `https://addi.ehu.es/handle/10810/19678`.

[Ministerio de Empleo y Seguridad Social, 2018] Ministerio de Empleo y Seguridad Social (2018). Boletín oficial del estado - xvii convenio colectivo estatal de empresas de consultoría y estudios de mercado y de la

opinión pública. Disponible en: https://www.boe.es/boe/dias/2018/03/06/pdfs/BOE-A-2018-3156.pdf.

[Muñoz, 2017] Muñoz, R. (2017). El gobierno confirma un ciberataque masivo a empresas españolas. Disponible en: https://elpais.com/tecnologia/2017/05/12/actualidad/1494585889_857386.html.

[Méndez, 2008] Méndez, G. (2008). Especificación de requisitos según el estándar de ieee 830. Disponible en: https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf.